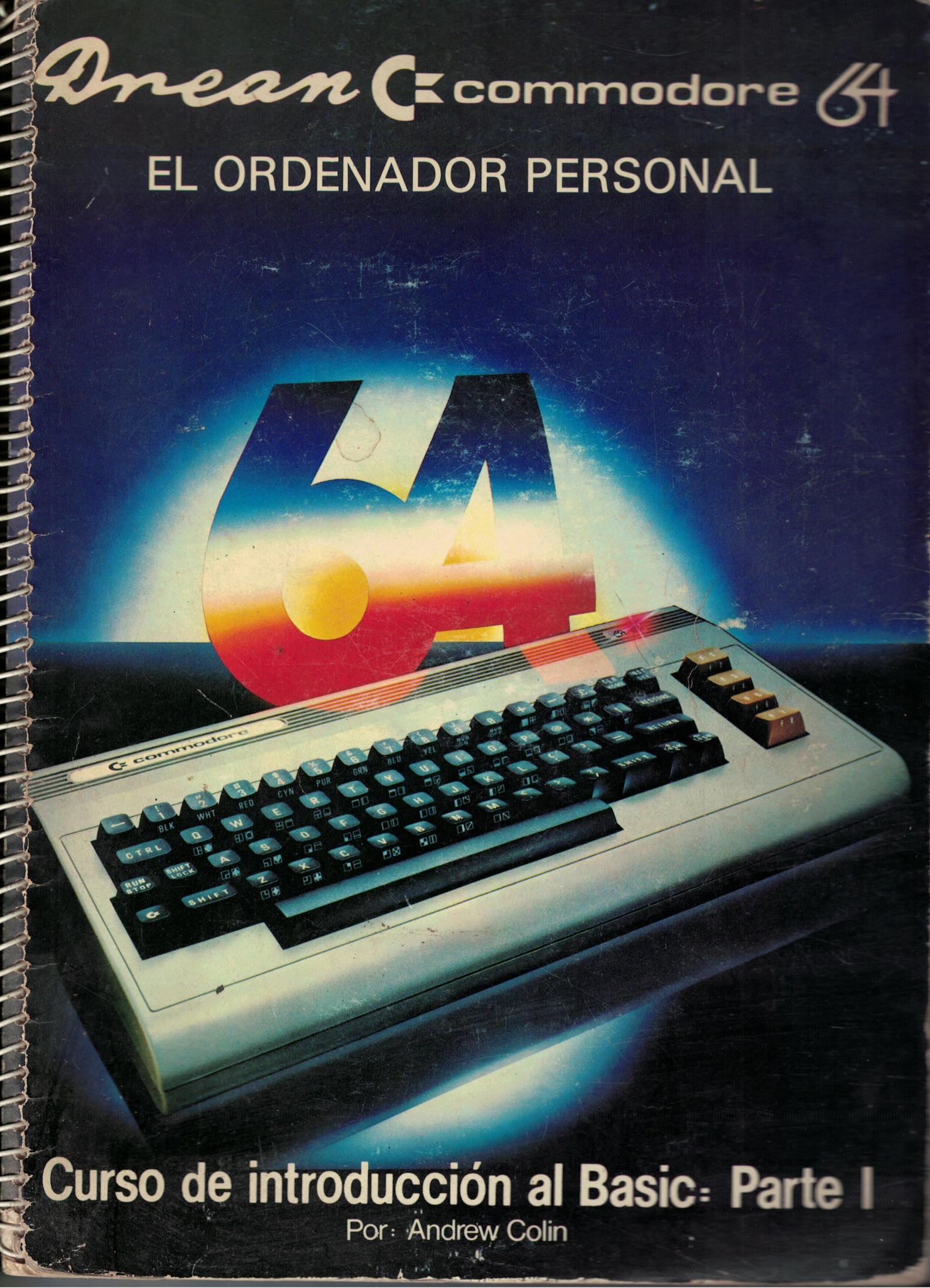
Dream commodore 64

# EL ORDENADOR PERSONAL




# Curso de introducción al Basic: Parte I

Por: Andrew Colin

# INTRODUCCION

Le damos nuestra cordial bienvenida a este curso de programación para el C-64*. El C-64, además de ser un avanzado y completo ordenador para todo tipo de aplicaciones, es ideal para la ejecución de programas recreativos y producir fascinantes figuras y sonidos en el receptor de televisión.

Los ordenadores son extraordinariamente versátiles, más que cualquier cosa, excepto una persona. Vd. puede usar el C-64 como máquina didáctica, calculadora, juegos, registro financiero, control de valores, monitor en una unidad de cuidados intensivos, controlador de un proceso industrial o procesador de datos técnicos o científicos, como el diseño de edificios, centrales eléctricas o aeronaves.

Los ordenadores y los sistemas que controlan, van introduciéndose a ritmo acelerado en todos los ámbitos de la actividad cotidiana. Numerosos dispositivos, tales como semáforos, máquinas registradoras y terminales de bancos utilizan ordenadores. Esta tendencia es cada vez más marcada; el mundo atraviesa una revolución de la informática, de consecuencias tan profundas como la propia Revolución Industrial lo fue en su día.

La revolución desencadenada por los ordenadores es imparable, pero de cada uno depende el ejercer cierta influencia sobre su evolución. Cada vez más, el mundo va dividiéndose en dos categorías de personas: los pasajeros y los pilotos. Los primeros sólo son espectadores, pueden disfrutar usando los dispositivos que usan productos del ordenador u odiar los ordenadores. Nunca llevan el control y no sabrían como llevarlo si pudieran. Los pilotos, por otra parte, llevan el control de esta revolución generalizada en todos los niveles de nuestra sociedad: inventan nuevos tipos de ordenadores y crean formas originales y más útiles de aplicarlos; su responsabilidad es grande, ya que tienen asignadas la tarea de desarrollar las aplicaciones de los ordenadores hacia la paz y la libertad, evitando una sociedad irracionalmente mecanizada como la que describe la ciencia ficción en sus obras más pesimistas.

¿Qué diferencia al piloto del pasajero? Simplemente el comprender la forma en que el ordenador funciona. Desde luego, hay diferentes niveles de conocimiento. La mayoría de la gente sabe cómo usar el juego de "los invasores del espacio", aunque no conozcan el mecanismo del ordenador. No obstante, el nivel de comprensión que sirve de base a esa diferencia es mucho más profundo: consiste en ser capaz de que un ordenador haga lo que uno quiera, tanto si se trata de actividades recreativas como didácticas, de aplicaciones industriales o médicas.

Para poder ejercer este dominio sobre un ordenador, para convertirlo en un servidor obediente y exacto, usted ha de saber programar la máquina. La programación es la clave para convertirse en piloto.

Este curso es sobre programación. Se refiere de un modo especial a los ordenadores COMMODORE C-64, pero es básicamente aplicable a otros ordenadores.

Cuánto más programe, más fácil le será el hacerlo. La mayor parte de la gente puede aprender a programar si se lo propone, y seguro que usted también puede hacerlo. No se necesita ningún conocimiento exhaustivo sobre matemáticas, pero es esencial poder concentrarse; busque un rincón tranquilo en su casa donde pueda leer, pensar y usar su ordenador.

Para sacarle el máximo rendimiento al Basic, es importante que Vd. avance paso a paso en el estudio y aplicación de este curso.

El curso se divide en quince unidades, cada una de las cuales suele requerir una o dos tardes de estudio. En cada unidad, hay una parte de lectura, otra de trabajo práctico con el C-64 y un poco de programación, así como un cuestionario que permite evaluar el nivel de comprensión de cada lección. Todas las unidades incluyen experimentos, que Vd. debe marcar a medida que los realiza.

Hay también preguntas que Vd. debe contestar, rellenando el espacio dejado a tal fin. Para ello, utilice el lápiz blando fácil de borrar, para que los ejercicios puedan ser realizados también por otra persona a quien un día puede entregar el curso. Si, por otra parte, este curso le ha sido dado a Vd. por otra persona que ya lo siguió, borre con una goma cuidadosamente todas las respuestas antes de iniciar el estudio.

Conceptos que se han aprendido en unidades anteriores, son mencionadas posteriormente, sin volver a ser explicadas. Por ejemplo, Vd. no entenderá la lección 10 si no ha estudiado detenidamente todas las anteriores. Es esencial que Vd. siga este curso en el orden que se presentan las materias.

Cuando empiece una nueva unidad, lea desde el principio al final sin detenerse en cada detalle, así se hará una idea de los temas

(*) En este libro "C-64" significa "COMMODORE 64", el ordenador sobre el que se basa este curso.

que se van a estudiar. El próximo paso es estudiar detenidamente, tratando de entender todo, lo fácil y lo que parece más difícil.

Programar puede aprenderse sólo practicando; Vd. no puede aprender un instrumento musical con sólo teoría. Es, por lo tanto, esencial que complete todos los problemas de programación del curso. Tan pronto como se sienta capaz, empiece a plantear sus propios problemas y a darles soluciones.

Cuando haya completado este curso, podrá usar el C-64 para múltiples aplicaciones: podrá realizar juegos inventados por Vd. mismo, resolver ecuaciones y todo tipo de cálculos. En estos programas podrá incluir imágenes de color diseñadas por Vd. mismo, así como sonido para realzar el significado, tanto tonos agradables como ruidos menos atractivos.

Sin embargo, la programación es un tema muy extenso y no sería posible aprender en un solo curso. Al cabo de cierto tiempo Vd. probablemente querrá ampliarlo; quizás para solventar problemas más complejos o usar su ordenador para controlar un tren eléctrico miniatura. Para estas aplicaciones, COMMODORE está desarrollando una serie de cursos de programación avanzada.

Pase ahora a la unidad de estudio 1. ¡Le deseamos suerte!

## NOTAS A LA UTILIZACION DE ESTE CURSO

Como complemento al CURSO DE INTRODUCCION AL BASIC PARTE I adjuntamos esta pequeña nota que queremos sirva como aclaración a posibles dudas que pudiera plantear la redacción de este Curso.

Cuando en este Curso se indica que debemos cargar un programa en el ordenador, quiere decir que tendremos que escribir este mismo programa conforme a los listados que se adjuntan en el Apéndice D.

¿Cómo entraremos un programa? Fíjese que al principio de cada línea de programa encontrará un número. Este número indica al ordenador el orden en el que tendrá que ejecutar las instrucciones que le entraremos. Entonces, y teniendo en cuenta esto, entraremos **SIEMPRE** en primer lugar el número de línea, luego las instrucciones que le sigan, y cuando vea que lo que sigue a la última instrucción es otro número de línea, deberá pulsar la tecla RETURN. Esto hará que lo que Vd. acaba de escribir quede dentro de la memoria del ordenador.

Una vez que tenga todo el programa en la memoria de su ordenador, le recomendamos que antes de ejecutarlo, lo guarde en algún periférico de almacenamiento de masa, como puede ser el Datassette (TM) o la unidad de discos VIC-1541.

¿Cómo guardarlo en una cinta virgen? Bien, asegúrese de que tiene el programa entrado en memoria. Escriba SAVE"nombre de programa" y pulse RETURN. El ordenador le contestará PRESS RECORD & PLAY ON TAPE. Pulse las teclas **RECORD** y **PLAY** en el magnetofón, se apagará la pantalla, y cuando ésta vuelva ya se habrá grabado el programa en la cinta. Pero aún falta otro paso: hay que verificar si está bien grabado.

¿Cómo lo haremos? Con el comando VERIFY. Escriba este comando, rebobine la cinta y pulse RETURN. Se apagará la pantalla de nuevo hasta que encuentre un programa, con lo cual volverá de nuevo la pantalla para anunciar FOUND nombre de programa, estará así durante unos segundos tras lo cual se apagará de nuevo la pantalla y verificará el programa. Si nos da OK no hay ningún problema. Si no, habría que volver a grabar este programa del modo en que lo hemos explicado antes.

Hay otro asunto. Una vez hemos guardado el programa, apagamos el ordenador y lo volvemos a encender, con la pérdida del programa que teníamos en memoria.

¿Cómo lo recuperaremos? Bien, como hemos tenido la precaución de guardar previamente el programa en cinta y ahora lo recuperaremos escribiendo LOAD"nombre de programa" y pulsando la tecla RETURN. El ordenador contestará PRESS PLAY ON TAPE y deberá pulsar la tecla PLAY del magnetofón. Una vez hecho esto, seguir el procedimiento que se seguía cuando hablábamos de VERIFY. Cuando se haya cargado el programa, escriba RUN y pulse RETURN.

¿Y por qué entrar el programa? Muy sencillo. Nosotros creemos que el hecho de entrarlo le va a ayudar a adquirir confianza en el manejo del ordenador. Le va a ayudar también a aprender a depurar programas (es muy raro que un programa funcione bien a la primera, incluso para los profesionales).

NOTA: sobre todo, en aras de que no se encuentre con problemas, le recomendamos que lea atentamente la Introducción que se hacen a los programas listados en el Apéndice D.

# CONTENIDO

## OBJETIVOS DE ESTE CURSO

Este curso constituye la Parte I de una Serie Didáctica concebida para ayudarle a aprender todos los aspectos de la programación del ordenador Commodore C-64. Este primer curso trata los principios de programación y los aspectos fundamentales del lenguaje BASIC, y consta de dos partes principales:

1. Un texto autodidáctico dividido en 15 lecciones o "unidades" de estudio, cada una de las cuales analiza un importante aspecto de la programación.
2. El listado de todos los programas a los que se hace referencia en el texto, tanto los ejemplos como los propios cuestionarios que le ayudarán a verificar sus progresos.

Este curso le enseñará como escribir programas de utilidad y entretenimientos para los ordenadores C-64, pero no abarca la totalidad del lenguaje BASIC. Los aspectos más avanzados del BASIC se tratan en la Parte II, o segundo curso, de esta Serie.

## INDICE DE MATERIAS


| Título | Temática | Programas listados en el apéndice D | Pág. |
|---|---|---|---|
| | Introducción | | |
| Unidad 1 | *Preparativos:* Conexión del Sistema C-64; carga de programas desde el cassette; ajuste del receptor TV | CARTA DE AJUSTE<br>AHORCADO | 1 |
| Unidad 2 | *El teclado:* el cursor, signos gráficos, dibujo de figuras; escritura en pantalla. | VELMEC | 7 |
| Unidad 3 | *Figuras en color:* control de los colores del fondo y marco de la pantalla; selección del color de los caracteres sobre fondo invertido | UNI3CUEST | 15 |
| Unidad 4 | *Comandos directos:* números y cadenas; el comando PRINT; efectos de la coma y punto y coma sobre el espaciado; variables; el comando LET; operadores aritméticos y de cadena. | UNI4EJER | 23 |
| Unidad 5 | *Comandos almacenados* (en reserva) programas almacenados; comando GOTO; bucles simples. (no controlados) | UNI5CUEST | 31 |
| Unidad 6 | *Ayudas prácticas:* el comando LIST; edición de líneas; grabación y verificación de programas; errores frecuentes. | FRASES | 37 |
| Unidad 7 | *Bucles controlados:* condiciones referentes a números y cadenas; control de bucle por contaje, etc.; el significado DE "=" en BASIC. | UNI7CUEST | 45 |

| Título | Temática | Programas listados en el apéndice D | Pág. |
|---|---|---|---|
| Unidad 8 | *Identificación:* localización de errores | UNI8PROG | 57 |
| Unidad 9 | *Colores programados:* modo de pantalla normal y especificado; representación en la pantalla de los caracteres de control; uso de los caracteres de control de posición y color en los programas; el reloj interno TI$. | UNI9CUEST | 65 |
| Unidad 10 | *Entrada de datos:* el comando INPUT; *relaciones entre el programador y* el usuario. | UNI10CUEST | 73 |
| Unidad 11 | *Diagramas de flujo:* comandos condiciones; validación de datos; diagramas de flujo; glosarios; diseño del programa. | UNI11PROG | 79 |
| Unidad 12 | *Control avanzado de bucles:* los comandos FOR y NEXT; estructura de los programas. | UNI12CUEST | 93 |
| Unidad 13 | *Sonidos:* las voces del C-64; control de altura, volumen y duración. | DEMOSON<br>GENSON | 101 |
| Unidad 14 | *Programas de reducción de datos:* terminación de un flujo de datos; robustez de los programas. | CARA/CRUZ | 107 |
| Unidad 15 | *Juegos:* tiempo de reacción; el comando GET: el cronometrador interno TI; la función RND; estructuración de juegos de azar. | REACCION<br>o | 117 |
| | Recomendaciones finales. | | |
| Apéndice A | Aspectos matemáticos del C-64: Expresiones<br>Precisión de cálculo<br>Funciones *normalizadas (standard).* | | 129 |
| Apéndice B | Solución de problemas | | 135 |
| Apéndice C | Errores frecuentes | | 149 |
| Apéndice D | Listado de los programas | | 151 |

# UNIDAD: 1


EXPERIMENTO 1.1 PAG. 3

EXPERIMENTO 1.2 4

En esta unidad se empieza el uso del C-64. Se exponen aspectos elementales de manejo del ordenador y cuestiones de tipo práctico que suelen plantearse al iniciar la operación de un ordenador personal.

Para estudiar programación conviene disponer de un ambiente cómodo y tranquilo y destinar períodos relativamente largos (por lo menos 2 horas) a ello en un momento del día en que no esté demasiado cansado para concentrarse. Adopte las medidas necesarias para que no le interrumpan, como descolgar el teléfono o poner un letrero en la puerta; sin concentración no se puede estudiar, y no hay nada que dificulte tanto la programación como las constantes interrupciones.

1

*Si usted ya tiene su ordenador C-64 instalado* y ha empezado, puede saltar esta introducción y pasar al experimento 1.1; por otra parte, no será una pérdida de tiempo el que *siga leyéndola, ya que puede encontrar alguna* observación útil.

Primero, sitúe el equipo y conéctelo a la red. El transformador especial y cassette C-64 deben colocarse en la mesa, frente a usted, y el receptor de televisión conviene que esté por lo menos a 2 metros de distancia, si su pantalla es pequeña, o incluso más alejado, si es grande. Las imágenes y texto generados por C-64 son suficientemente grandes como para ser vistos debidamente a distancia normal. El usar una pantalla grande demasiado cerca es molesto y cansa.

Conecte los diversos aparatos que componen el sistema C-64 tal como se indica en el esquema de esta página. Cuando enchufe clavijas y conectores, ejerza una presión firme, pero suave; no aplique nunca excesiva fuerza; en vez de ello, fíjese en la disposición de las clavijas y receptáculos, para que coincidan sin presión excesiva.

El C-64 es una máquina extremadamente resistente, pero las clavijas y receptáculos pueden desgastarse o dañarse si se manipularon con excesiva frecuencia. Procure dejar las diferentes partes del C-64 interconectadas siempre que sea posible.

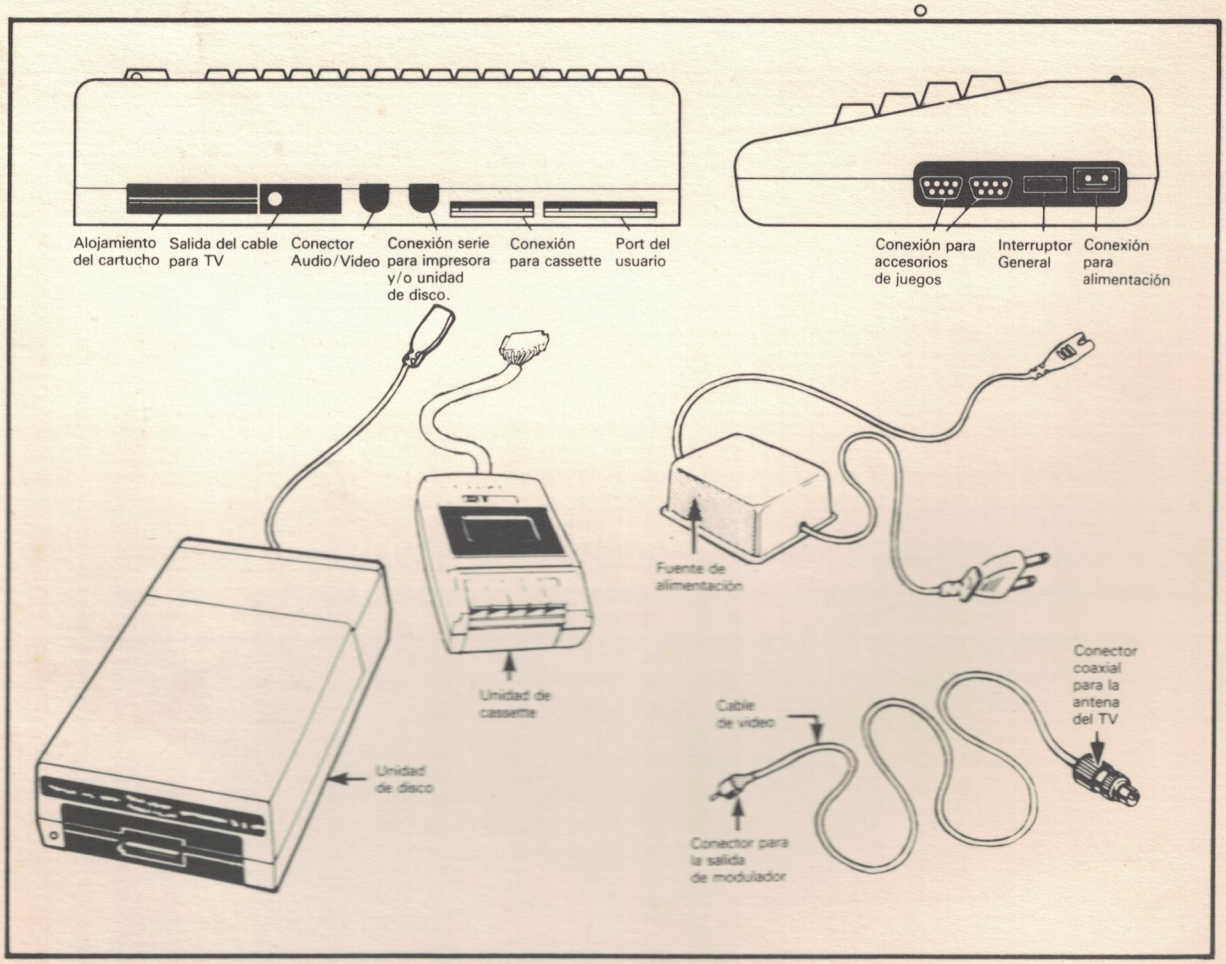
Si usted emplea un receptor de televisión que se destina también como televisor normal, intercale un conmutador de antena, para que *no tenga que desenchufar constantemente una* y otra antena.

Es práctico enchufar el ordenador C-64 y el receptor de televisón en un mismo cable de *extensión, con dos tomas de corriente en su* extremo. Este tipo de conexión confiere mayor movilidad al sistema C-64, para que usted pueda situar cada componente en el lugar y posición más favorables.

Conecte el sistema de la siguiente forma:

Primero conecte el televisor y seleccione un canal que normalmente no se use para programas televisivos. Por ejemplo, si usa el canal nº 1 para el primer programa y el nº 2 para el segundo programa, cualquier otro canal disponible será idóneo para conectar al C-64. En un principio el receptor hará mucho ruido y seguramente tendrá que reducirlo con

o

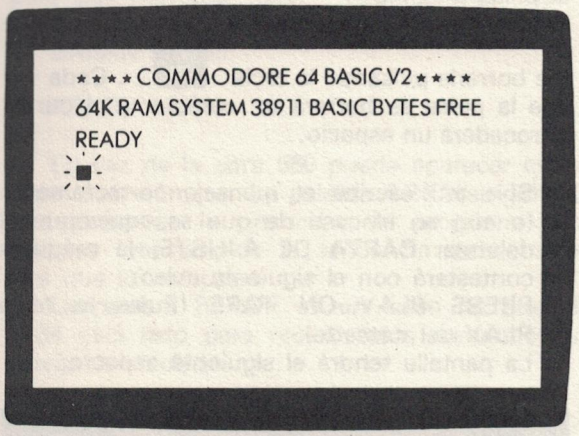


el mando de volumen.

Ahora conecte el ordenador a la red, accionando el interruptor situado en el panel derecho. Si todo está bien, se iluminará la lámpara piloto roja (POWER); sin embargo; en la pantalla no aparece normalmente todavía imagen.

Vuelva ahora a su receptor de televisión y gire a derecha e izquierda el mando de sintonizar. El procedimiento de sintonización varía según la marca del televisor y se explica en las instrucciones del fabricante; muchas veces se trata de girar a mano o con un destornillador un diminuto mando situado detrás de una ventanilla en el panel frontal. Si tiene que usar un destornillador, no toque ninguna otra pieza aparte del mando de sintonía en cuestión, ya que de lo contrario podría sufrir una descarga eléctrica.

Cuando gire el mando de sintonía del televisor en una u otra dirección aparecerá repentinamente en la pantalla la siguiente imagen:




Dentro de la pantalla aparece un rectángulo azul oscuro enmarcado por un borde color azul claro. Para que la imagen aparezca inmóvil tendrá que ajustar seguramente los mandos de sincronismo horizontal y sincronismo vertical.

Si esta imagen no aparece o lo hace sólo en blanco y negro, desconecte el C-64 accionando su interruptor de conexión-desconexión y vuelva a conectarlo al cabo de unos segundos.

Si se presenta algún problema, compruebe los siguientes puntos:

- Asegúrese de que el televisor funciona correctamente, conectando uno de los canales normales para el primero o segundo programa. De no funcionar, hágalo reparar.
- Si no se ilumina la lámpara piloto POWER, compruebe:
  (a) Si hay corriente en la línea;
  (b) Que otro aparato eléctrico p.e. secador pelo, funciona en el mismo enchufe. Si no funciona cambie el fusible que afecta a esa extensión;
  (c) Si está intacto el fusible de la clavija del transformador especial (ensaye cambiando el fusible);
  (d) Que el C-64 está correctamente enchufado.
- Asegúrese de que el C-64 está enchufado en el correcto receptáculo de antena del televisor.

Si, a pesar de realizar todas esas comprobaciones, el sistema sigue sin funcionar, póngase en contacto con su distribuidor para repararlo.

En la pantalla del televisor aparecerá una serie de letras, números y símbolos. Estos caracteres son siempre del mismo tamaño y su número máximo, con la pantalla completa, es de 1.000.

La primera línea del mensaje que ha aparecido en la pantalla identifica el producto como un ordenador programable en BASIC fabricado por CBM (Commodore Business Machines). V2 representa el número del módulo o versión que puede variar, ya que periódicamente CBM produce variaciones más avanzadas del producto.

En la segunda línea del mensaje se indica la cantidad de memoria disponible, expresada en "bytes". Esta unidad de memoria es la requerida para alojar un símbolo o carácter de información. Cuanto mayor es la capacidad de memoria de un ordenador, más complejas pueden ser las tareas a realizar por la máquina.

Al conectar su ordenador COMMODORE 64 aparecerá en pantalla: 64k RAM 38911 BASIC BYTES FREE READY. Esto indica que el ordenador está esperando alguna instrucción.

Si por otra parte, el número de bytes es inferior a 38911 o difiere del valor real, es indicio de que el C-64 está averiado, en cuyo caso debe llevarlo al servicio técnico del distribuidor para su reparación.

En la tercera línea del mensaje apareció la palabra READY. Ello indica que C-64 está listo para recibir sus instrucciones, a mecanografiar en el teclado.

Por último, en la pantalla se ve un pequeño cuadrado centelleante. Este se llama "cursor" y cuando usted escribe en el teclado se desplaza para indicar en cada momento la posición en que aparecerá el siguiente carácter. Por ejemplo, escriba:

PRINT 5+8 y pulse la tecla RETURN

(Ello exige 10 pulsaciones, a saber:

PRINT SPACE 5+8 RETURN ). Antes de empezar a escribir en el teclado, es conveniente comprobar la tecla SHIFT LOCK para asegurarse de que no esté en posición apretada. Cada vez que usted pulsa una tecla, con excepción de

RETURN , en la pantalla aparece un símbolo y el cursor avanza un espacio. La función primordial de la tecla RETURN es hacer que el ordenador ejecute la instrucción mecanografiada en el teclado; en este caso, la instrucción (PRINT) consistía en exhibir el resultado de sumar 5+8.

# EXPERIMENTO 1·1



Para que un ordenador pueda realizar cualquier tarea debe incorporarse en un programa y cargarse en la memoria. Los programas pueden almacenarse en cintas cassettes, discos o cartuchos. A continuación se describe cómo cargar un programa desde una cinta cassette a la memoria del ordenador. Supongamos que tiene grabado en cinta un programa que se llama CARTA DE AJUSTE y luego otro llamado AHORCADO (ambos listados se dan en el apéndice D), aunque el ejemplo se da a continuación sirve para cualquier programa en cinta, para ello sólo debe cambiarse "CARTA DE AJUSTE" por el nombre del programa. En caso de no conocer dicho nombre teclee "" en lugar de "CARTA DE AJUSTE". Siga estas instrucciones cuidadosamente:

1. Asegúrese de que el cassette esté enchufado al C-64.
2. Pulse la tecla STOP del cassette.
3. Abra la tapa del cassette, saque cualquier cinta que pueda haber y coloque la cinta CARTA DE AJUSTE, con dicha etiqueta hacia arriba y la ventanilla de visión de la cinta hacia usted. Cierre la tapa. Si ésta no se cerrara completamente, hasta quedar plana, no la fuerce, sino asegúrese de que la cinta está correctamente insertada.
4. Pulse la tecla REWIND del cassette. Observe la cinta por la ventanilla: si la cinta está rebobinando espere hasta que se pare en el principio.
5. Pulse la tecla STOP del cassette.
6. Ahora escriba el siguiente mensaje en el teclado.

LOAD "CARTA DE AJUSTE" RETURN

Ello exige 22 pulsaciones en total, contando cada signo comillas como 1 pulsación. Para obtener el símbolo comillas, apriete una de las dos teclas SHIFT y, sin soltarla, pulse " 2 . Suelte la tecla SHIFT tan pronto (pero no antes) como el signo " aparezca en la pantalla. Es esencial que el anterior mensaje se mecanografíe correctamente. A continuación se explican algunos errores frecuentes en la pulsación de esas teclas:

Escribir con la tecla SHIFT LOCK presionada. Ello da orígen a una serie de extrañas líneas, corazones y otros signos, sin congruencia alguna.

Usar dos apóstrofes en vez de comillas. En tal caso, el C-64 pondrá en pantalla el siguiente mensaje de error:

?SYNTAX
ERROR
READY.

No es necesario que corrija el error. Simplemente vuelva a empezar a mecanografiar el mensaje en la línea siguiente.

Pulsar la barra espaciadora entre las comillas y CARTA, o entre AJUSTE y las comillas de cierre. Pulsar las teclas R E T U R N en vez de la tecla. RETURN . En este caso no ocurre nada.

Pulsar la tecla cero (ϕ) en vez de la letra O en la palabra LOAD.

Si usted pulsa una tecla errónea, puede siempre borrarla pulsando la tecla INST DEL . Cada vez que la pulse se borrará un carácter y el cursor retrocederá un espacio.

7. Si usted escribe el mensaje correctamente (o aún en el caso de que se equivoque al deletrear CARTA DE AJUSTE, la máquina contestará con el siguiente aviso:
   PRESS PLAY ON TAPE (Pulse la tecla PLAY del cassette).
   La pantalla tendrá el siguiente aspecto.

Pulsar, pues, la tecla PLAY y esperar un poco mientras el programa se carga en la memoria del ordenador.

Si la cinta no para de girar y en la pantalla aparecen varios mensaje tales como

FOUND CARTA DE AJUSTE
FOUND AHORCADO

es probable que usted haya escrito el nombre incorrectamente. Pare el ordenador pulsando la tecla RUN STOP y vuelva a repetir el procedimiento a partir de la operación nº. 1.

Si la cinta no se parara ni apareciera mensaje alguno, compruebe que no se trate de una cinta vacía. De lo contrario podría ser que el cassette estuviera averiado, por lo que debería

llevarlo al distribuidor para su comprobación.
Cuando el programa esté cargado en memoria, aparecerá el siguiente mensaje en la pantalla:

READY.

Entonces puede ejecutarse el programa escribiendo R U N y pulsando RETURN (en total 4 pulsaciones).

El programa, en su ejecución, muestra la gama de colores seleccionables con el C-64 y es idóneo para que usted realice ajustes de precisión en su receptor de televisión.

Gire el mando de volumen para incrementar el sonido y ajuste al canal con precisión, hasta que la música sea clara y con el mínimo ruido de fondo. Entonces ajuste los mandos de luminosidad y color, de modo que los colores correspondan a sus nombres y sean correctos vistos desde cierta distancia.

Cuando quiera parar la ejecución del programa CARTA DE AJUSTE, pulse la tecla RUN STOP.

Al pulsar esta tecla (o cuando el programa para por cualquier razón), en la pantalla aparece el siguiente mensaje:

BREAK IN 560

READY

En vez de la cifra 560 puede aparecer cualquier otro número. La palabra BREAK significa una pausa en la ejecución del programa por haberse interrumpido la secuencia de comandos que lo compone.

La palabra READY es un aviso de que el C-64 está listo para recibir otros comandos a través del teclado.

Si el sonido continúa, apriete la tecla RUN STOP y, antes de soltarla, pulse la tecla RESTORE ; ello tendrá siempre como efecto la interrupción del sonido y simultáneamente el borrado de la pantalla.

En ocasiones, cuando el programa CARTA DE AJUSTE se para a mitad de una nota, ésta sigue sonando. Para silenciarla, apriete las dos últimas teclas citadas.

La ejecución del programa CARTA DE AJUSTE es especialmente útil para ajustar el receptor de televisión, por lo que puede cargarse en memoria siempre que se precise dicho ajuste.

*Fin del Experimento 1.1*

# EXPERIMENTO 1·2



El experimento 1.2. es un juego para adivinar palabras concebido para familiarizarse con el teclado. Cargar el programa AHORCADO.

```
**** COMMODORE 64 BASIC V2 ****
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY
LOAD "AHORCADO"
PRESS PLAY ON TAPE
```

Cuando el programa esté correctamente cargado en la memoria del ordenador y aparezca el mensaje READY, escriba

RUN y pulse la tecla RETURN

El juego empezará. Si usted no sabe cómo jugar, pulse letras al azar y observe (y escuche) lo qué ocurre. Pronto comprenderá de qué se trata.

Practique este juego todo el tiempo que quiera y adquiera la oportunidad de familiarizarse con el teclado.

*Fin del Experimento 1.2*

# UNIDAD: 2


| | | |
|---|---|---|
| EXPERIMENTO 2.1 | PAG. | 7 |
| EXPERIMENTO 2.2 | | 8 |
| EXPERIMENTO 2.3 | | 10 |
| EXPERIMENTO 2.4 | | 11 |

# EXPERIMENTO 2·1

Esta unidad de estudio trata del teclado y tiene como fin enseñar a utilizarlo para escribir mensajes y dibujar figuras en la pantalla.

Si usted ha utilizado antes una máquina de escribir, verá que el teclado es similar. Las letras, números y otros signos están en los sitios de costumbre. Hay también teclas para seleccionar mayúsculas (SHIFT) y para retenerlas (SHIFT/LOCK), aún cuando en el C-64 sus funciones sean algo diferentes.

Por otra parte, si nunca ha usado una máquina de escribir no se preocupe: evidentemente, requerirá un poco más de tiempo para acostumbrarse al teclado, pero es la única diferencia.

En el curso de esta unidad de estudio, como excepción, no use la tecla RETURN a menos que se lo indiquemos. Como ya vió en la lección anterior, esta tecla es la que determina la acción del ordenador, como por ejemplo cargàr un programa en la memoria o realizar una suma. Como nuestro objetivo ahora es simplemente usar la pantalla, no necesitamos realmente la intervención del ordenador. Si usted pulsa de hecho RETURN el C-64 tratará de obedecer el mensaje o figura que se haya puesto en pantalla, con el resultado de que los interprete erróneamente y estropee lo escrito.

Otra tecla que no debe pulsar durante esta unidad de estudio es la del signo comillas ("). Este signo tiene un significado especial y altera la forma en que la pantalla reacciona a otras teclas. Si en la pantalla aparece el signo comillas, puede que sea mucho más difícil dibujar figuras que tengan sentido. La aplicación de la tecla comillas se explica más adelante, en otra lección; por ahora, evite pulsarla.

Puede que encuentre difícil abstenerse de pulsar las teclas citadas, pero no se preocupe: el C-64 no incluye ningún comando que pueda destruirlo, contrariamente a los ordenadores descritos en la Ciencia Ficción. Usted, no puede causar ningún daño al C-64 pulsando teclas erróneas. Algunos caracteres asociados con comillas o la tecla RETURN pueden hacer que se comporte de modo muy extraño, y ciertas secuencias de pulsaciones, que usted podría ejecutar casualmente, enmudecerán completamente a su amigo el ordenador; sin embargo, estas excentricidades o enfados tienen sólo carácter temporal y usted puede superarlos desconectando el ordenador y volviéndolo a conectar al cabo de 30 segundos.

Las 66 teclas (del teclado) se dividen en dos categorías:

- 50 teclas de signos y caracteres que usted tiene a su disposición para escribir letras, números y signos gráficos;
- 16 teclas de funciones, que controlan las letras, números y signos.

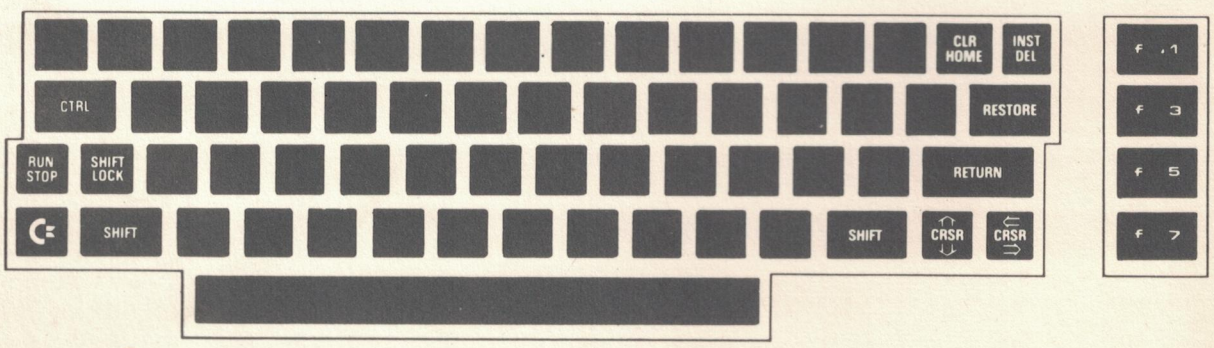
Las teclas de funciones son: CTRL, RUN STOP, SHIFT LOCK, C=, SHIFT, CLR HOME, INST DEL, RESTORE, RETURN, CRSR ↓, CRSR →, SHIFT y f1 a f8

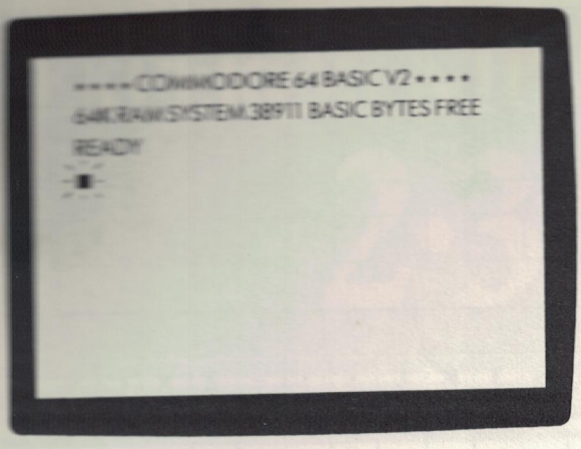
Las teclas de números, del 1 al Ø, tienen además ciertas funciones de control.

Compare el teclado del C-64 con el siguiente diagrama e identifique las teclas de funciones.

Pulse la tecla SHIFT LOCK varias veces: verá que tiene dos posiciones, una apretada y otra elevada. Al terminar, asegúrese de que está en la posición elevada.

Conecte el C-64 de la forma acostumbrada. Debajo del mensaje READY, verá el cursor centelleante.






# EXPERIMENTO 2·2



En este Experimento examinaremos el movimiento del cursor en la pantalla a medida que se escribe en el teclado. La finalidad del cursor es señalar dónde va a aparecer el siguiente carácter que usted va a pulsar. Escriba, pues, unas cuantas letras y observe como el cursor se desplaza hacia la derecha. Vea como cada nuevo carácter sustituye al cursor, que entonces pasa a la siguiente posición.

Ahora llene toda la línea con letras, hasta que el cursor quede arrinconado junto al borde derecho. Pulse entonces una letra más: el cursor salta al comienzo de la próxima línea, sin necesidad de hacer nada para cambiar de línea.

Antes de continuar, cuente las letras que caben en una línea y anote su número, en lápiz, en la siguiente casilla:

Hay ☐ espacios para caracteres en cada línea de la pantalla.

Continúe mecanografiando líneas hasta que llegue a la última que cabe en la pantalla. Entonces cuente las líneas y anote su número en la siguiente casilla; no se olvide de contar las líneas en blanco que hay encima y debajo del mensaje:

38911 BYTES FREE

Hay ☐ líneas de caracteres cuando la pantalla está llena.

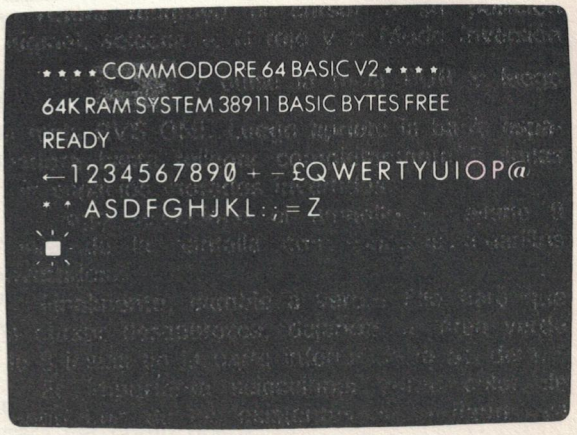
Ahora complete la última línea ya empezada hasta que el cursor quede arrinconado en la esquina inferior derecha de la pantalla. Pulse un nuevo carácter y observe el cursor: la totalidad de la pantalla se desplaza hacia arriba, para dejar sitio a una nueva línea, en cuyo principio se sitúa el cursor. Esta nueva línea en blanco se ha obtenido en sustitución de la línea superior, que ha desaparecido. Las líneas superiores que desaparecen de esta forma son irrecuperables, a menos que se copien en alguna otra parte.

Mecanografíe unas cuantas líneas más, para comprobar que efectivamente vuelve a surgir una línea adicional en blanco en la parte inferior de la pantalla cada vez que la capacidad de ésta se agota.

*Fin del Experimento 2.1* ☐

El C-64 tiene unas 50 teclas de signos, muchas de ellas dobles: incluyen letras mayúsculas y minúsculas, números, signos de puntuación, símbolos matemáticos y una amplia gama de signos gráficos y elementos simples que pueden combinarse en todo tipo de figuras. Para obtener determinados signos y caracteres es necesario pulsar simultáneamente la tecla SHIFT (una cualquiera de las dos marcadas con esta inscripción) y la tecla especial Commodore, marcada C=.

Para realizar el Experimento 2.2, vuelva a conectar el ordenador y escriba:
← 1 2 3 4 5 6 7 8 9 0 + - £ Q W E R T Y U I
(es decir, todas las teclas de signos de la hilera superior y algunas de la segunda hilera). La pantalla adquirirá el siguiente aspecto:




Ahora mantenga apretada una de las SHIFT y vuelva a pulsar las mismas teclas: en la pantalla aparecerá una línea casi completamente diferente (incluyendo el símbolo comillas, pero ello no planteará problemas si sigue las instrucciones). Copie ahora los símbolos que acaban de aparecer en la segunda hilera de la tabla de la pág. 9; verá que algunos de los signos gráficos coinciden con las letras (por ejemplo en el caso de la U y la I). Si su televisor lo permite, los signos aparecerán como enmarcados en las propias teclas. Vea como los signos gráficos tienden a tocar los

| SIMBOLO | ← | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Ø | + | – | £ | Q | W | E | R | T | Y | U | I |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SHIFT | | | | | | | | | | | | | | | | | | | | | | |
| C= | | | | | | | | | | | | | | | | | | | | | | |

| SIMBOLO | O | P | @ | ★ | ↑ | A | S | D | F | G | H | J | K | L | : | ; | = | Z | X | C | V | B |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SHIFT | | | | | | | | | | | | | | | | | | | | | | |
| C= | | | | | | | | | | | | | | | | | | | | | | |

| SIMBOLO | N | M | , | . | / | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SHIFT | | | | | | | | | | | | | | | | | | | | | | |
| C= | | | | | | | | | | | | | | | | | | | | | | |

pequeños recuadros que ocupan, con lo cual se pueden usar para formar figuras compactas

Luego pulse por tercera vez las mismas teclas, pero esta vez manteniendo la tecla C= apretada. Muchos de los signos vuelven a ser diferentes. Copie, pues, esos signos en la tercera hilera de la tabla.

Para ver el aspecto que adquiere en la pantalla el resto de los signos, repita el experimento con las otras dos líneas del teclado.
O P @ ★ ↑ A S D F G H I J K L : ; = Z X C V B
N M , , /

Complete esta última línea con espacios.

Recuerde que Ø es siempre el dígito cero, mientras O la letra "o".

Pulse simultáneamente C= y SHIFT. Muchas de las mayúsculas cambiarán a minúsculas. Vuelva a pulsar a la vez ambas teclas y reaparecerán las mayúsculas. En general puede usarse un juego completo de signos gráficos o bien un juego limitado y minúsculas, pero no ambos juegos simultáneamente. El uso de minúsculas se explica en el segundo volumen de este curso.

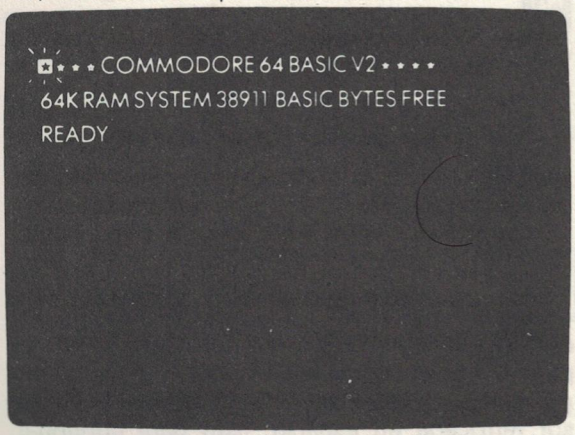
*Fin del Experimento 2.2*

# EXPERIMENTO 2·3

En los anteriores ejercicios le hemos pedido que visualizara caracteres en la pantalla simplemente en orden consecutivo, de izquierda a derecha, en líneas completas. Ahora le enseñaremos cómo colocar dichos caracteres en cualquier punto específico de la pantalla.

A este fin se usan tres teclas de control del cursor: CLR HOME, CRSR ↓, y CRSR →.

Cuando usted pulsa CLR HOME, el cursor vuelve a su posición original, que es la esquina superior izquierda de la pantalla. Conecte el ordenador, o desconéctelo y vuélvalo a conectar si acaba de realizar el Experimento 2.2, en el caso de que algún comando siguiera ejerciendo su acción. Al conectarlo, el cursor aparece debajo del mensaje "READY". Al pulsar CLR HOME, retrocede a su posición superior izquierda, superpuesto al signo estrella del extremo izquierdo; la razón de que la estrella continúe visible

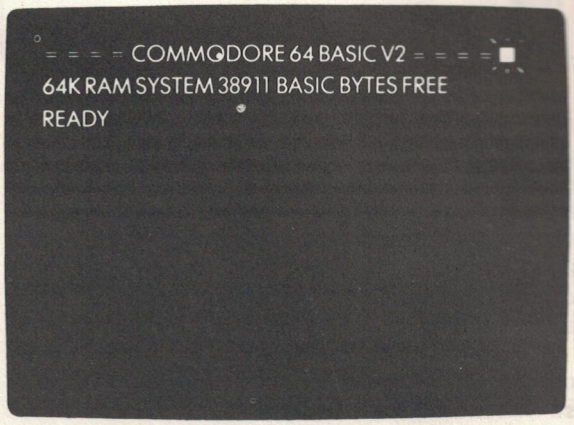



es que el cursor es "transparente"; sin embargo, si usted pulsa cualquier tecla de caracter (o la barra espaciadora) el signo visible debajo del cursor será sustituido por el nuevo carácter. Por ejemplo, sustituya la estrella por el signo "igual a" (=). Púlselo tres veces más. La visión de la pantalla será:

= = = = COMMODORE 64 BASIC V2 ★ ★ ★ ★

Supongamos que ahora usted quiere cambiar los símbolos de estrella de la derecha también por cuatro "=", para que la línea sea simétrica. Sin embargo, si usted pulsa la barra espaciadora para pasar a la primera estrella, se encontrará con que el título va borrándose cada vez que pulsa el espaciador. Para evitarlo, pulse la tecla CRSR →: el cursor avanza como usted desea, sin borrar el mensaje ya existente.

Sitúe de este modo el cursor sobre la primera estrella de la derecha y pulse cuatro veces el signo =; la línea superior de la pantalla adquirirá el siguiente aspecto:




Si en vez de pulsar repetidamente la tecla CRSR → usted la mantiene apretada, el cursor se desplazará a una velocidad de 10 espacios por segundo; al llegar al final de la línea, pasará al comienzo de la siguiente, y así sucesivamente.

Si antes de pulsar CRSR → mantiene apretada la tecla SHIFT, el cursor se desplazará en sentido contrario, es decir, retrocederá al principio de línea y luego subirá al final de la línea anterior.

Ensaye esta función para situar el cursor al principio de la primera línea y trate de sustituir el signo = por el signo de estrella, para dejar dicha línea tal como estaba antes.

Baje luego el cursor a la última línea de la pantalla y vea qué ocurre si va más allá del final de línea: la pantalla se desplaza hacia arriba, como ocurrió antes de pulsar un carácter adicional.

Pulse CLR HOME para restituir el cursor a su posición original y trate de hacerlo retroceder: la pantalla no baja, si es lo que usted esperaba; no ocurre nada y el cursor permanece inmóvil.

La tecla CRSR ↓ se usa para subir y bajar el cursor una línea. Practique un poco su uso, para ver cómo opera.

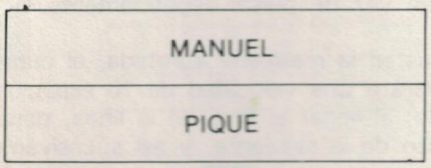
Ahora escriba unos cuantos caracteres y signos gráficos y pulse de nuevo CLR HOME, pero esta vez manteniendo la tecla SHIFT apretada: no sólo el cursor vuelve a la esquina superior izquierda, sino que la pantalla se borra, dejándola lista para empezar de nuevo.

Rellene la siguiente tabla:

| Tecla | Efecto | |
|---|---|---|
| | Sin shift | Con shift |
| CLR HOME | Cursor posición original | |
| CRSR ⇐⇒ | | El cursor retrocede 1 espacio |
| CRSR ⇑⇓ | | |

Practique ahora dibujando en la pantalla con signos gráficos y las teclas de control del cursor. Empiece con figuras geométricas simples, tales como cuadrados y otros cuadriláteros, triángulos y pequeñas circunferencias. Cuando se equivoque, retroceda el cursor y pulse la tecla correcta. Para borrar caracteres que estén en lugar erróneo pulse la barra espaciadora.

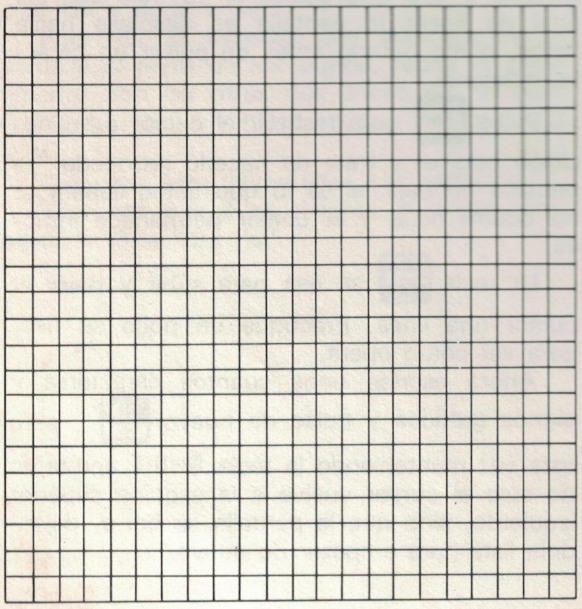
Cuando haya practicado un poco, dibuje dos rectángulos como los siguientes y escriba su nombre en su interior.




Ahora dibuje algunas cartas de jugar, con esquinas redondeadas y las figuras correctas (escoja cartas de figuras simples).

Finalmente, si se ve con ánimos y tiene afición al dibujo, trace figuras tales como algún animal, una nave espacial o las facciones de una persona.

Esta tarea será más fácil si hace primero un esbozo con lápiz en la siguiente cuadrícula.



*Fin del Experimento 2.3*



# EXPERIMENTO 2·4

Todo el mundo hace faltas al mecanografiar. Si usted pulsa una letra errónea en medio de una palabra, puede corregirla con las teclas de control del cursor: simplemente sitúe el cursor sobre dicha letra y pulse la correcta; por ejemplo, si ha escrito AUSTRAPIA, retroceda el cursor hasta que esté sobre la P y pulse la L. Haga algunas correcciones para ver lo fácil que es.

Si en cambio, tiene demasiadas o insuficientes letras en una palabra, no puede usar este método. Para suprimir una letra que haya pulsado de más o intercalar otra, debe usar la tecla INST DEL

Al pulsar INST DEL, desaparece el carácter existente a la *izquierda* del cursor y el resto de las letras retrocede un espacio para rellenar el que había quedado vacío.

Por ejemplo, si ha escrito INXDIA, para borrar la X sitúe el cursor sobre la D y pulse INST DEL. La X desaparece y DIA retroceda un espacio para dejar la palabra correcta INDIA.

Practique este uso de INST DEL para corregir errores de mecanografía como los siguientes:

| | |
|---|---|
| CHINNA | = CHINA |
| EEGIPTO | = EGIPTO |
| FINLANDIIA | = FINLANDIA |
| AUSTRALIA | = AUSTRIA |

El uso más frecuente de INST DEL es borrar el carácter o caracteres erróneos acabados de pulsar. La razón de ello es que con sólo una **pulsación se borra el último símbolo y el cursor** retrocede un espacio, listo para continuar. Uno se acostumbra pronto a pulsar INST DEL cada vez que se comete un error de mecanografía.

La otra función de INST DEL, consistente en intercalar espacios en medio de palabras o líneas. Se obtiene apretando una de las teclas SHIFT antes de pulsar INST DEL. Estos espacios pueden entonces rellenarse con caracteres en la forma acostumbrada.

Ensaye esta función, cambiando la palabra AUSTRIA en AUSTRALIA.

Para ello, borre primero la pantalla ( SHIFT + CLR HOME ) y escriba:

AUSTRIA

Retroceda el cursor para que quede situado sobre la I.

Apriete la tecla SHIFT y, antes de soltarla, pulse dos veces INST DEL : con cada pulsación, la terminación IA avanzará un espacio hacia la derecha, mientras que el cursor permanecerá parado. De este modo, puede pulsar directamente AL en los dos espacios que se intercalaron. Avance luego el cursor hasta que rebase la palabra.

Para practicar el uso de la tecla INST DEL , borre de nuevo la pantalla y escriba la siguiente lista de palabras de la izquierda, y luego conviértalas a las de la derecha:

| | |
|---|---|
| HOTEL | MOTEL |
| MICROFONO | MICROORDENADOR |
| MADRAS | MADERAS |
| ANDAS | ANDARAS |
| BARCA | BARCAZA |
| PELA | PELOTA |
| SOBREMESA | MESA |
| CALAR | CALLAR |
| CORRER | RECORRER |
| PALMA | PALOMA |
| TACTO | TACITO |
| LACRA | LACA |
| BUENA | HIERBABUENA |
| LIRA | LIBRA |
| MALA | ALA |
| CASO | CASADO |
| TRUCO | TRUCADO |
| PERSA | PERSIANA |
| CARRETA | CARA |
| PASCAL | PASCUAL |
| MARIO | MARIANO |
| JAMAICA | JAVA |

Al terminar, vuelva a cambiar las palabras a su forma original.

*Fin de Experimento 2.4*

El programa de la Unidad 2 lleva el título VELMEC. Su finalidad es familiarizarse con el teclado. Cárguelo en la memoria y ejecútelo tantas veces como le parezca necesario para practicar.

# UNIDAD: 3


| | | |
|---|---|---|
| EXPERIMENTO 3.1 | PAG. | 15 |
| EXPERIMENTO 3.2 | | 16 |
| EXPERIMENTO 3.3 | | 18 |
| EXPERIMENTO 3.4 | | 20 |

El C-64 es un ordenador de color. En esta unidad de estudio se introducen algunas de las formas con que Vd. puede hacer que el C-64 dibuje figuras multicolores en la pantalla del televisor.

Evidentemente, si usted usa un televisor en blanco y negro no puede esperar resultados brillantes; sin embargo, realice los experimentos descritos a continuación.



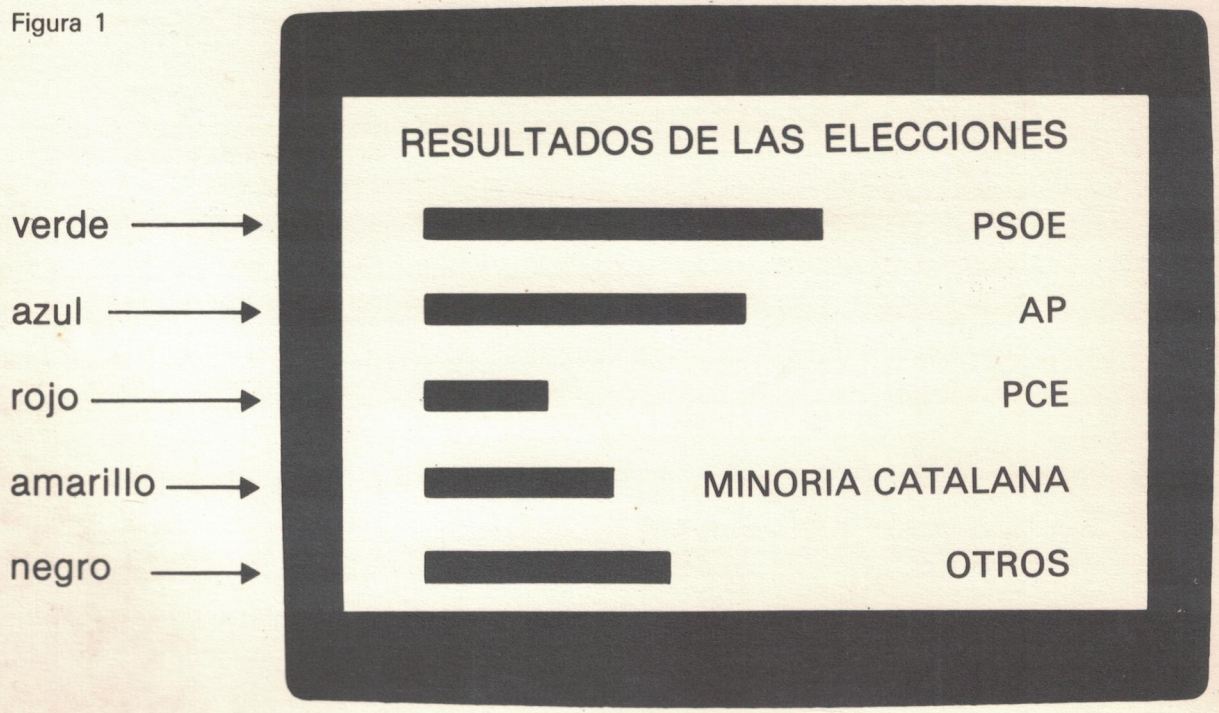
# EXPERIMENTO 3·1

Cargue y ejecute el programa CARTA DE AJUSTE (Unidad 1) para asegurarse de que el receptor de televisión esté correctamente ajustado. Para parar el programa, apriete RUN STOP y pulse la tecla RESTORE

El cursor es ahora de color gris. Usted puede cambiar su color, que será también el que adoptarán los caracteres o signos gráficos que entonces pulse. Ensaye con una serie de signos (apriete C= y pulse la tecla I), que aparecerán en gris, el color actual del cursor.

(Cuando un carácter tenga detalles muy finos, puede que el color no sea a veces correcto. Ello ocurre cuando se usa un receptor de televisión corriente, que posee una banda de radiofrecuencias estrecha. Ello podría superarse adquiriendo un monitor de televisión más costoso, pero a menos que se quiera visualizar una gran cantidad de texto en diferentes colores, ello no vale la pena).

Figura 1



El color del cursor puede cambiarse en cualquier momento pulsando una de las 8 teclas de colores, mientras se mantiene apretada la tecla CTRL . Las teclas de colores —que son también las de los número 1-8— llevan su color indicado con las abreviaciones inglesas.

Apriete CTRL y pulse sucesivamente las diferentes teclas de colores:

Cuando pulsa la tecla 1 (BLK = black) el cursor cambia a negro.

Cuando pulsa la tecla 7 (BLU = azul) el cursor desaparece; la razón de ello es que ahora es del mismo color que el fondo de la pantalla y, por consiguiente invisible.

Las otras teclas cambian el cursor sucesivamente a rojo, cián, púrpura, verde, blanco y amarillo.

También puede seleccionar otra gama de colores utilizando la tecla COMMODORE y la tecla de color.

Así tendremos los siguientes colores:

C= + BLK = Naranja

C= + WHT = Marrón

C= + RED = Rosa

C= + CYN = Gris oscuro

C= + PUR = Gris medio

C= + GRN = Verde claro

C= + YEL = Gris claro

Practique diferentes colores de visualización dibujando imágenes coloreadas diferentemente. Por ejemplo, trace barras de diferentes colores y longitudes. Use a este fin el signo gráfico ■ ( C= + U) para construir las barras. Para trazar una barra roja de 5 signos de longitud, apriete CTRL y pulse la tecla 3 (RED): el cursor cambiará a rojo; entonces apriete C= y pulse U cinco veces.

Cuando haya adquirido cierta práctica con las teclas de colores 1-8, trate de dibujar un diagrama de resultados electorales locales, como el ilustrado en la fig. 1. Mecanografíe las letras en negro, para que los colores de las barras resalten más.

*Fin de Experimento 3.1*

# EXPERIMENTO 3·2



Además de conferir diferentes colores a cada signo o grupo de signos, usted puede seleccionar los colores del borde y del fondo de la pantalla. A este fin C-64 no posee teclas exclusivas, como en el caso de los colores de los signos o caracteres; en vez de ello debe usarse el comando POKE.

Escriba, por ejemplo, POKE 53280,0. Asegúrese de que ha pulsado los números correctamente; si es necesario, corríjalos. Luego pulse la tecla RETURN Inmediatamente después el borde de la pantalla se convierte al color negro.

La instrucción utilizada consta de tres partes:

POKE : el comando o palabra clave.

53280 : es una "dirección", porque identifica una posición de memoria de C-64 que determina los colores del borde de la pantalla; cualquier instrucción que se use para cambiar el color de uno u otro elemento debe citar siempre esta dirección.

Ø : es un código que indica "borde negro".

Para seleccionar otros colores y combinaciones conviene conocer los códigos de colores. Hay 16 colores, cada uno con su correspondiente número de identificación; cada uno de estos 16 puede asignarse al fondo de la pantalla.

| *Color* | *Número* | *Observaciones* |
|---|---|---|
| Negro | 0 | |
| Blanco | 1 | Pueden usarse para el fondo, el borde y los caracteres. Sus números o códigos son el número indicado en su pertinente tecla, menos 1. |
| Rojo | 2 | |
| Cián | 3 | |
| Morado | 4 | |
| Verde | 5 | |
| Azul | 6 | |
| Amarillo | 7 | |
| Naranja | 8 | |
| Marrón | 9 | |
| Rosado | 10 | Para obtener caracteres en estos colores se debe usar la tecla C= |
| Gris oscuro | 11 | |
| Gris medio | 12 | |
| Verde claro | 13 | |
| Azul claro | 14 | |
| Gris claro | 15 | |

El color del fondo de la pantalla es seleccionado de la misma forma que el del borde pero en la dirección 53281.

Pruebe la siguiente combinación:

POKE 53280,4

POKE 53281,11

Seleccione unas cuantas combinaciones de colores con el comando POKE.

Seleccione los colores del fondo y borde de la pantalla que más le agraden.



*Fin del Experimento 3.2*

# EXPERIMENTO 3·3

Quizás a usted le habrá parecido que faltan algunos signos gráficos de importancia básica; por ejemplo, tenemos el signo pero no el ; tenemos  y pero no y

Además, parece imposible rellenar todo un cuadrado con color a fin de construir áreas grandes de la misma tonalidad.

Para obtener estos signos tan necesarios nos valemos del dispositivo de inversión de campo, que invierte los colores del carácter y del fondo. Este proceso de inversión cromática se pone de manifiesto en el siguiente experimento:

Conecte el C-64, seleccione morado como el color del cursor y escriba unos cuantos caracteres, incluyendo , , y un espacio. Ahora apriete CTRL y pulse la tecla 9 (que lleva también el rótulo RVS ON).

Luego suelte la tecla CTRL y escriba unos cuantos caracteres más: aparecerán en azul, sobre un fondo morado, en vez de morado sobre azul. El cambio se hace más evidente en que cambia a , y en y , que cambian a y , mientras el espacio aparece como un bloque sólido de morado. A esta modalidad de representación de los signos y caracteres se denomina "modo invertido".

Para cambiar a escritura normal, apretar CTRL y pulsar la tecla Ø (que lleva también el rótulo RVS OFF).

Con sólo mirar el cursor no puede deducirse si el ordenador está en Modo Invertido o Modo Normal. En caso de que use muchos signos gráficos invertidos y con el continuo cambio se olvide del modo de operación, puede solventar esta dificultad con las dos formas siguientes:

(a) Escriba el siguiente carácter y obsérvelo: si es erróneo, bórrelo, cambie el modo y púlselo de nuevo.

(b) Pulse la tecla RVS ON o RVS OFF (previamente apriete CTRL ) antes de escribir el siguiente carácter. Si la máquina ya estaba en el modo correcto ello no ejercerá ningún efecto.

La mejor forma de dibujar bloques de color es con la pulsación de la barra espaciadora en el Modo Invertido. La barra espaciadora es repetitiva, y si usted la mantiene apretada generará una serie de espacios con una velocidad de 10 por segundo.

Una buena manera de practicar el uso de la barra espaciadora para obtener áreas de colores es el dibujo de banderas nacionales de diseño horizontal, como la bandera española. Ensayemos con la bandera del SUDAN, que tiene tres colores:

| morado |
|---|
| amarillo |
| verde |

Primero, seleccione un color adecuado para el borde de la pantalla (que se convertirá en el marco de la bandera), como por ejemplo el negro. Como fondo seleccionaremos el mismo color que la franja inferior de la bandera, o sea el verde. La instrucción será:

POKE 53280,0 RETURN

POKE 53281,5 RETURN

Ahora restituya el cursor a su posición original, seleccione el rojo y el Modo Inversión (apriete CTRL y pulse la tecla PUR y luego la tecla RVS ON). Luego apriete la barra espaciadora hasta rellenar completamente 8 líneas con espacios morados invertidos.

Luego seleccione el amarillo y rellene 9 líneas de la pantalla con espacios amarillos invertidos.

**Finalmente, cambie a verde. Ello hará que el cursor desaparezca, dejando un área verde de 8 líneas en la parte inferior de la bandera.**

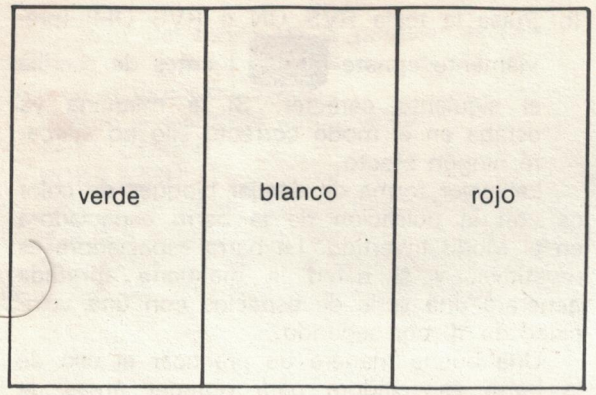
Es importante seleccionar como color de fondo uno de los existentes en la figura, ya que de lo contrario no se podría ocultar el cursor al final del dibujo. De igual modo, el color del borde debe ser diferente de todos los que aparezcan en la figura.

Cuando sepa construir sin problemas banderas horizontales, practique con otras verticales o en cruz, entre éstas últimas la suiza y la irlandesa.

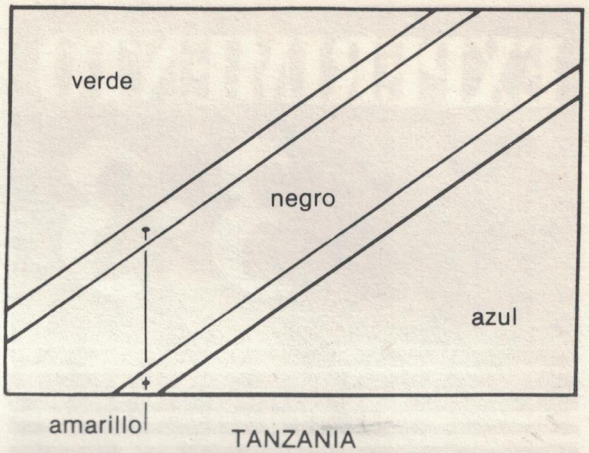
Las banderas con franjas diagonales, como las de Tanzania y Checoslovaquia, son naturalmente más difíciles. Las partes de la bandera junto a las líneas oblicuas deben construirse con los signos gráficos o



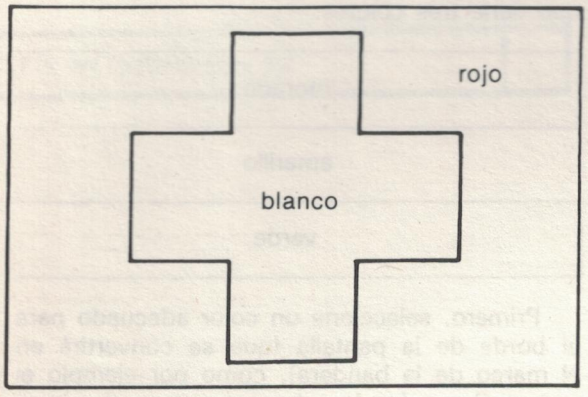
Dream Commodore



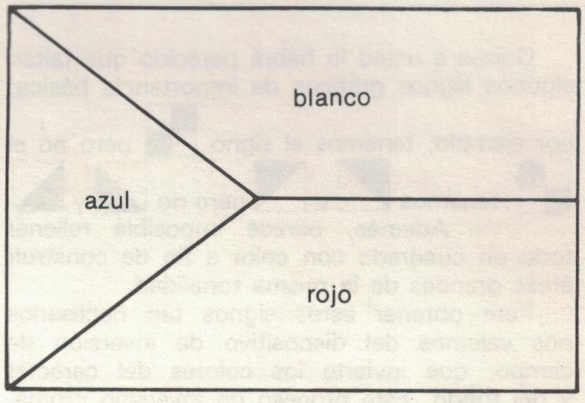

ITALIA



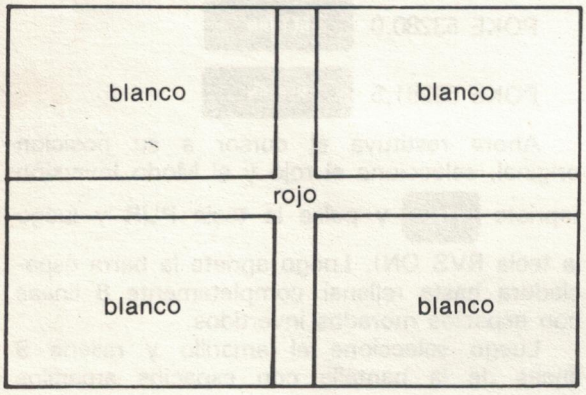

TANZANIA



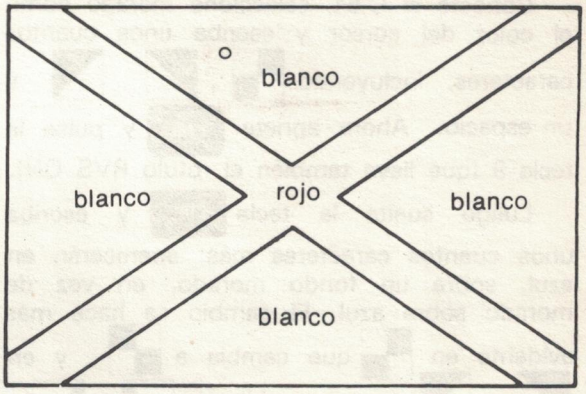

SUIZA



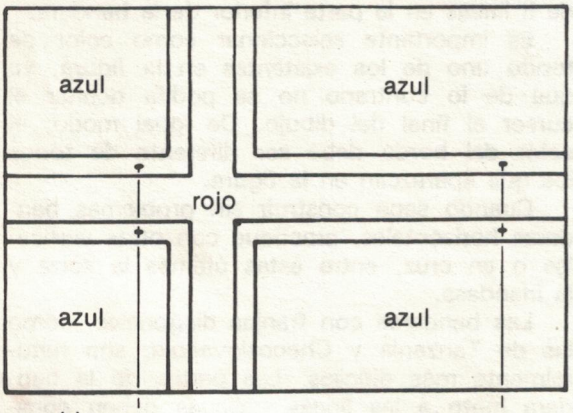

CHECOSLOVAQUIA



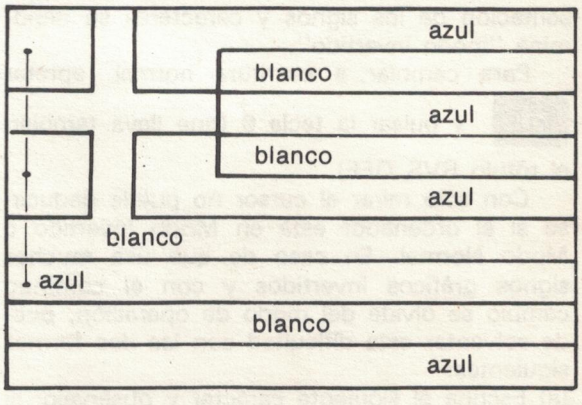

SAN JORGE




SAN ANDRES



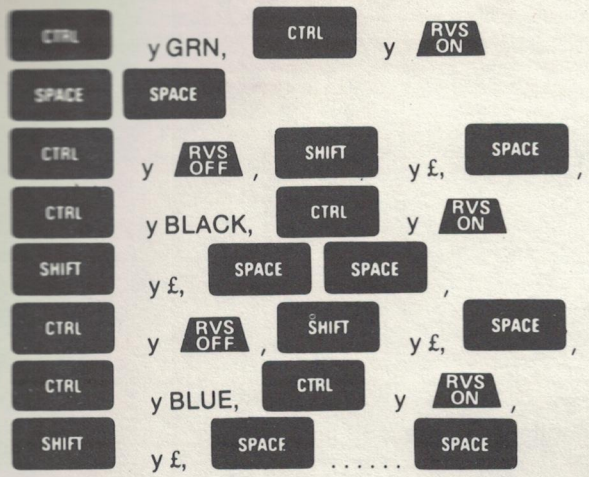

ISLANDIA




GRECIA

de ser necesario invertidos. En estos casos, naturalmente, debe seleccionarse un color de fondo que aparezca al efectuar la inversión junto a las líneas oblicuas. En el caso de la bandera de Tanzania, el fondo podría ser amarillo: no podría usarse en cambio el azul, ya que no habría forma de dibujar un elemento verde y amarillo.

Para trazar una línea oblícua en la bandera de Tanzania, se podría proceder de la siguiente forma:




Naturalmente, las comas y la conjunción "y" se incluyen sólo para facilitar la escritura de las instrucciones y no deben pulsarse.

*Fin de Experimento 3.3*

# EXPERIMENTO 3·4



Trate de hacer sus propias obras de arte usando toda la gama de colores del C-64. Probablemente le será útil plantear primero su dibujo en un papel cuadriculado, y usar lápices de colores.

*Fin del Experimento 3.4*

El programa que va con la Unidad 3, es un test que puede ser cargado en memoria entrando el programa que se lista en el apéndice D.

# UNIDAD: 4


EXPERIMENTO 4.1 PAG. 23

EXPERIMENTO 4.2 26

En las tres primeras unidades de estudio del presente curso nos hemos concentrado casi exclusivamente en el teclado del C-64 y en su uso para visualizar texto y figuras. Esta preparación es esencial para pasar a la siguiente parte del curso, en que tratamos algunas de las funciones programables.

Para que el C-64 realice tareas es necesario indicárselo utilizando palabras clave, que se llaman comandos o instrucciones y se expresan en BASIC, un lenguaje sencillo para ordenadores desarrollado originalmente por Kemeny y Kurtz en el Dartmouth College de Estados Unidos. El BASIC tiene también reglas de gramática, como todo lenguaje, pero son fáciles de aprender y no tardan en retenerse en la memoria al practicarse su uso.



Todas las instrucciones del BASIC empiezan con una palabra clave, como por ejemplo LOAD, POKE o PRINT, y terminan con la pulsación de la tecla RETURN

La tecla RETURN tiene dos aplicaciones:

Si la palabra clave no lleva números antepuestos al principio de línea, dispone la ejecución inmediata del comando. Por ejemplo, si usted escribe:

LOAD "CARTA DE AJUSTE"

la carga del programa "CARTA DE AJUSTE" empezará cuando se pulse RETURN

El otro significado de RETURN se trata en la siguiente unidad de estudio, pero como referencia recuérdese que si la palabra clave al comienzo de línea lleva antepuesto un número, como por ejemplo

30PRINT 5 + 7

el efecto de RETURN no es ejecutar el comando sino almacenarlo en la memoria como parte integrante de un programa, para su posterior ejecución.

En esta unidad de estudio nos limitamos a la primera aplicación de RETURN es decir los comandos no llevarán números antepuestos y serán ejecutados inmediatamente después de pulsarse dicha tecla.

# EXPERIMENTO 4·1

Uno de los comandos más útiles y polifacéticos es PRINT; dispone la ejecución de todo tipo de operaciones y la visualización de sus resultados en la pantalla. La razón de que se use el término "print" para designar la puesta en pantalla, cuando en realidad su significado es "imprimir", es que originalmente, cuando se creó el lenguaje BASIC, en la Universidad de Dartmouth, la salida del ordenador era una teleimpresora.

El Experimento 4.1 consta de tres etapas: primero se ensayan varios comandos PRINT, cuyos resultados anotamos. Luego explicamos las características del comando que se han puesto de manifiesto en los ejemplos ejecutados. En la tercera etapa tratamos de predecir lo que el ordenador hará en respuesta a determinadas entrada; estas respuestas pueden entonces comprobarse usando el ordenador.

Empiece, pues, escribiendo las siguientes instrucciones terminando cada una con RETURN

Asegúrese de que ha mecanografiado correctamente cada instrucción; si hay algún error, corríjalo con las teclas de control del cursor. Las respuestas que vaya obteniendo anótelas en las correspondientes casillas de la siguiente tabla. Las dos primeras casillas ya están rellenadas, a título de ejemplo.

| | |
|---|---|
| PRINT999 | 999<br>READY. |
| PRINT"HOLA" | HOLA<br>READY |
| PRINT—56 | |
| PRINT3 + 4 + 4 | |
| PRINT5★7 | |

| | |
|---|---|
| PRINT 27/7 | |
| PRINT "ORDENADOR COMMODORE 64" | |
| PRINT C-64 | |
| PRINT3,5 | |
| PRINT3;5 | |
| PRINT"CASA","PISO" | |
| PRINT"GATO","PEZ" | |
| PRINT"3+5" | |
| PRINT29—12;"LEONES" | |
| PRINT1;2;3;4 | |

Observe ahora atentamente las notas que usted ha realizado y trate de detectar las diferencias que aparecen en las respuestas del C-64 según la forma con que se escribió el comando PRINT. En primer lugar vemos una característica común a todas las instrucciones: la palabra o mensaje READY aparece en cada caso, indicando que el operador puede continuar, pero ello no es una característica exclusiva de PRINT, sino de todos los comandos que determinan una visualización directa.

He aquí las diferencias o características principales de distintas versiones del comando PRINT:

1. PRINT puede manipular tanto "números" como "cadenas", si bien los trata diferentemente. Los números pueden mecanografiarse explícitamente (como 999) o en forma de expresiones que el ordenador debe calcular. Ejemplos de expresiones en nuestro experimento fueron 3+4+5,5★7, 27/7 y 29—12, o sea que vemos que el C-64 puede sumar, restar, multiplicar y dividir. El signo★ significa multiplicación y el signo /, división (en realidad, el C-64 puede realizar los cálculos más complejos que Vd. pueda necesitar, incluyendo paréntesis para entrar conjuntos y subconjuntos y toda clase de funciones de cálculo científico; vea el Anexo A, en que se relacionan estas posibilidades de cálculo). Una "cadena" es una serie de caracteres o números entre comillas. El comando PRINT muestra en pantalla tales cadenas exactamente tal como se escriben en el teclado, sin calcularlas. Las cadenas en nuestro caso eran: "HOLA","ORDENADOR C-64", "CASA","PISO","GATO","PEZ","3+5" y "LEONES".

   Así pues, vemos que "3+5" es una cadena, en vez de una expresión, al ir entrecomillada. Si usted quiere visualizar una cadena, pero se olvida de entrecomillarla, probablemente la respuèsta será Ø (aunque en ocasiones el resultado es otro).

2. Vemos también que el comando PRINT puede tratar dos o más cantidades o cadenas simultáneamente. Si las dos están separadas por comas, el segundo resultado aparece separado, empezando a mitad de la pantalla; si se usa punto y coma (;) la separación es menor, y en general, las cadenas no se separan, (así obtenemos GATOPEZ). Los números visualizados aparecen en cambio invariablemente separados en sus sucesivas cantidades; cada una de ellas va precedida de un espacio o del signo —, de ser negativa; hay también siempre un espacio al final de cada cantidad. Si no ocurrió así en su caso, repita el comando.

   PRINT1;2;3;4

   Para estar seguro de que hay un espacio entre cada número, recórralos con el cursor; verá que hay un espacio de cursor entre cada cantidad.

3. Consideremos el espacio de las letras y otros caracteres que integran el comando. La palabra clave propiamente dicha, PRINT, debe ser compacta, sin intercalar espacios, y cualquier espacio que se introduzca dentro de una cadena formará parte de la misma y será reproducido. Los espacios entre diferentes cadenas o cantidades, en cambio, se ignoran. Así, si escribimos

   PRINT 3 +5;7; 8

   obtenemos la misma visualización en la pantalla que si mecanografiamos

   PRINT3+5;7;8

   Esta regla —que los espacios en los comandos se ignoran a menos que estén dentro de una palabra clave o de una cadena— es válida para todo el lenguaje BASIC.



Dream Commodore

4. Si usted comete un error de mecanografía al escribir una palabra clave o si entra una explicación que no tiene sentido (como 5★★7), el ordenador las rechazará con un mensaje de error:

?SYNTAX

ERROR

Este mensaje indica que usted ha infringido alguna regla de BASIC. Ello no tiene importancia, simplemente corrija el error y continúe.

Ahora trate de predecir lo que el C-64 mostrará en pantalla como respuesta a los siguientes comandos. En la columna "Respuesta prevista" escriba en lápiz los resultados que usted prevee, con el espaciado que supone correcto; el espaciado es tan importante como adivinar el resultado en sí. Por otra parte, no repita cada vez la palabra READY.



Algunos de los comandos contienen quizás errores a propósito. En tal caso entre el mensaje que Vd. supone aparecerá.

Compruebe luego si sus predicciones eran correctas mecanografiando los comandos en el teclado y anotando cuidadosamente la visualización de C-64 en cada caso, en la columna "La respuesta del C-64". Si hay alguna diferencia que usted no puede explicarse, repita todo el Experimento desde un principio, hasta que todos los aspectos del comando PRINT le resulten claros.

Recuerde que el C-64 realiza siempre las multiplicaciones y divisiones antes que las sumas y las restas. Por ejemplo:

PRINT5+2★7 dará 19

y

PRINT5+2+6/3−3 dará 6.

| *Comando* | *Respuesta prevista* | *La respuesta del C-64* |
|---|---|---|
| PRINT 94 | | |
| PRINT8—5 | | |
| PRINT3★2+5 | | |
| PRINT"CARAMBA" | | |
| PRINT BASTANTE | | |
| PRINT"1★/3" | | |
| PRINT3;47 | | |
| PRINT2+2;2−2;2★2;2/2 | | |
| PRINT"LIMPIA";"CRISTALES" | | |
| PRINT8—7 | | |
| PRINT"18","RATON" | | |
| PRINT53+★7 | | |
| PRINT−1;−2;−3 | | |

*Fin del Experimento 4.1*

# EXPERIMENTO 4·2

Si los ordenadores pudieran realizar los comandos sólo de uno en uno carecerían de utilidad; a lo sumo, serían como calculadoras no programables. La mayoría de tareas realizadas con ordenadores se basan en completas secuencias de comandos, llamadas "programas". A medida que los comandos van ejecutándose, la marcha del programa y la transferencia de los resultados de un comando al siguiente se efectúa mediante porciones de memoria que adquieren la forma de variables.

Antes de pasar a hablar de las variables BASIC; consideremos una analogía humana. Supongamos que a usted le piden que lleve la cuenta de los goles de un partido de fútbol; las instrucciones podrían ser, por ejemplo:

Antes de que empiece el partido, dibuje dos casillas, cada una designada con el nombre de un equipo, y en su interior escriba ceros, o sea:

| RUMANIA | HUNGRIA |
|---|---|
| 0 | 0 |

Cada vez que uno u otro equipo marque un tanto, sustituya el último número escrito en su casilla por el mismo número más 1 y borre el número anterior.
En la media parte, el resultado podría ser:

| RUMANIA | HUNGRIA |
|---|---|
| ~~0~~ 2 | ~~0 1 2 3 4~~ 5 |

Cuando suene el silbato final, use el marcador para exhibir los nombres de ambos equipos y su respectivo tanteo:

| | |
|---|---|
| RUMANIA | 2 |
| HUNGRIA | 5 |

En la analogía de nuestro ejemplo, las casillas hacen las veces de variables; los números en las casillas registran el estado actual del juego en cada momento y cambian según los goles en el curso del partido; las instrucciones para usar las casillas son muy simples, pero a prueba de errores.

La memoria de C-64 (generalmente 38911 bytes, como se dijo antes) opera como un gran marcador. Cuando la máquina se conecta, el marcado se borra completamente; luego, cada vez que se cita una variable, se prepara una casilla, asignando cierta memoria a la misma, así como un nombre seleccionado por el usuario. Luego se entra un número en la casilla, para almacenar en ella el valor asignado.

El comando BASIC que indica al ordenador que debe realizar todo eso es la palabra clave LET. Examinemos el siguiente comando:

LET X = 5

En este ejemplo, el nombre de la variable es X. El C-64 prepara una localización (casilla) denominada X (de no haberla ya) y pone en su interior el número 5. Si ya existe una variable X, simplemente el número 5 sustituye el valor antes asignado a la misma. Vea los siguientes casos:

| | X no existe (Caso 1) | X ya existe (Caso 2) |
|---|---|---|
| Antes | (Memoria vacía) | 37<br>X |
| Después | 5<br>X | 5<br>X |

*Resultado de LET X = 5*

Cuando usted escribe un comando LET, la respuesta del ordenador en la pantalla es simplemente:

READY.

No hay ningún indicio en la pantalla de lo que ha hecho el ordenador. Afortunadamente, podemos utilizar el comando PRINT para que nos muestre el valor de la variable, siempre que citemos su nombre. Ensaye la siguiente secuencia de comandos:



Dream Commodore

| | *Resultados obtenidos* |
|---|---|
| LET Z = 14<br>PRINT Z | |
| LET Z = 31<br>PRINT Z | |

Al mecanografiar en el teclado esas instrucciones, el primer valor de Z que aparezca en la pantalla será 14, y el segundo, 31. El primer comando LET crea una variable llamada Z, a la que asigna el valor 14; la segunda instrucción simplemente cambia su valor a 31.

Conviene ahora exponer algunas reglas sobre las variables y sus nombres.

En el lenguaje BASIC hay dos clases de variables:

- Variables numéricas, para almacenar números, y
- Variables alfanuméricas, para almacenar cadenas (palabras o frases).

La selección de nombres para las variables es limitada: las variables numéricas pueden ser designadas por una sola letra, una letra seguida de un dígito o por dos letras; ejemplos:

A, X, Z, B5, TX, PQ

Los nombres de variables alfanuméricas terminan siempre con el signo $, pero por lo demás las reglas de su designación son idénticas a las variables numéricas; ejemplos de variables de cadena:

C$, Z$, P7$, DB$

Para ilustrar el uso de variables alfanuméricas, escriba en el teclado:

LET T$ = "BUEN"

LET T$ = "DIA"

El valor escrito a continuación del signo = en una instrucción LET no ha de ser necesariamente un número o cadena simple; puede ser una expresión o incluso el valor actual de otras variables que se citen con su nombre. Por ejemplo, examine los dos comandos siguientes:

LET Q = 5

LET S = Q+3

La primera instrucción crea una variable llamada Q (es decir, se trata de una variable numérica, al no llevar el signo $), cuyo valor se fija en 5. La segunda instrucción construye una variable denominada S, cuyo valor es el asignado a Q más 3. Para mejor comprender este tipo de operaciones, mecanografíe los dos anteriores comandos y luego ponga el resultado en la pantalla escribiendo:

PRINT Q;S

Ahora estudie las tres siguientes secuencias de instrucciones y anote en las casillas previstas a tal fin los resultados que usted prevee:

LET AA = 15

LET BB = 33—AA

PRINT AA,BB

LET D = 3

LET E = D★D+7

LET F = E—D

PRINT F;E;D

LET F = 4

LET F = F+1

PRINT F

Compruebe los resultados escribiendo las anteriores instrucciones en el teclado. ¿Ha obtenido todos los resultados correctos? ¿Incluso el último?

No hay límite al número de diferentes valores que puede alojarse en una variable, si bien en todo momento puede almacenarse sólo uno. En el comando de la última secuencia de instrucciones.

LET F = F+1

se elabora primero la expresión (tomando el valor de F y añadiendo 1) y el resultado se asigna a la localización F, sustituyendo el valor anterior. Es decir, el comando hace que el C-64 sume 1 al valor actual de F.

Los signos que nos permiten combinar números en diferentes formas se denominan "operadores" aritméticos". Son +, —, ★ y /. El BASIC permite también la manipulación de cadenas usando operadores de cadenas. Sólo uno de ellos combina dos cadenas; se denomina "concatenación" y se escribe con el signo +. Este operador simplemente une la segunda cadena al final de la primera, de modo que

"SOBRE" + "MESA" = "SOBREMESA"

Observe la siguiente secuencia de comandos y trate de predecir el resultado en la pantalla al escribir los comandos PRINT. Después mecanografíe la completa secuencia para comprobar si los resultados que usted ha anotado en las dos casillas corresponde a lo observado en la pantalla. No se olvide de pulsar la barra espaciadora antes de cerrar las comillas:

LET B$ = "PERRO SPACE "

LET C$ = "MUERDE SPACE "

LET D$ = "GATO SPACE "

LET E$ = B$+C$+D$

LET F$ = D$+C$+B$

PRINT E$

PRINT F$

PRINT y LET son los dos comandos más frecuentes del lenguaje BASIC. Sin embargo, en vez de escribir PRINT puede abreviarse a un interrogante (?), mientras que LET puede simplemente omitirse. Por ejemplo, una secuencia de ambos comandos perfectamente válida es:

```
A = 5

B = 17

?A,B
```

El programa que acompaña esta unidad de estudio está concebido para realizar prácticas con los comandos PRINT y LET. lleva por título UNI4EJER.

Cuando usted esté seguro de comprender plenamente el uso de variables numéricas y alfanuméricas puede parar la ejecución del programa.

*Fin del Experimento 4.2*

# UNIDAD: 5


| | | |
|---|---|---|
| EXPERIMENTO 5.1 | PAG. | 31 |
| EXPERIMENTO 5.2 | | 33 |

# EXPERIMENTO 5·1



Es hora de que empecemos a tratar el uso de comandos almacenados en la memoria antes de su ejecución, es decir programados. Veamos cómo el C-64 almacena estos comandos para su ulterior ejecución.

Conecte su ordenador (o si ejecuta algún otro programa, como por ejemplo VELMEC, párelo pulsando la tecla RUN STOP ) y escriba el comando NEW (seguido, como de costumbre, por la pulsación de la tecla RETURN ).

El comando NEW borra la memoria del C-64 de manera análoga a como el maestro borra la pizarra al principio de una lección. Evidentemente, usted no presenciará la acción de borrado, que se realiza en la memoria interna del C-64; la única respuesta que se hará visible será el mensaje READY.

Ahora escriba el siguiente comando, etiquetado con un número previo:

```
1Ø PRINT 13 + 59
```

Termine la línea pulsando la tecla RETURN . El único resultado visible es que el cursor pasa al comienzo de la siguiente línea. El resultado de la suma 13+59 (o cualquier otra) no aparece en la pantalla; en realidad, no se ejecuta la suma todavía, sino que la instrucción se almacena en la memoria interna del C-64.

Para comprobar esto, borre la pantalla (pulsando CLR HOME , previo apriete de SHIFT ) y escriba el comando:

```
LIST
```

Si usted ha pulsado todas las teclas correctamente, en la pantalla aparecerá una copia del comando etiquetado con un número; es una prueba de que se conserva en la memoria, a pesar de ser invisible en la pantalla.

Hemos realizado dos cosas: hemos almacenado nuestro comando PRINT en la memoria del ordenador y luego lo hemos exhibido en la pantalla; sin embargo, no ha sido ejecutado, es decir aún no sabemos el resultado de la suma de 13+59. Para averiguarlo, escribimos:

```
GOTO 1Ø
```

Goto indica al C-64 que ejecute la línea de programa específica. Por consiguiente, en la pantalla aparecerá inmediatamente el resultado de la suma.

Usted puede listar —obtener la lista— y ejecutar líneas de programa con LIST y GOTO tantas veces como quiera, sin que los comandos entrados en la memoria sufran deterioro.

El C-64 puede recordar simultáneamente un gran número de comandos (sólo limitado por la capacidad de memoria, ya que se requiere 1 byte por cada carácter de comando más un byte por cada comando). Para ello, cada línea del programa debe "etiquetarse", es decir debe asignársele un número de línea diferente. La máquina entra en memoria y lista los comandos por orden consecutivo de sus números de línea, en sentido ascendente, y los ejecuta también en dicho orden, a menos que se especifique lo contrario.

Ensaye ahora la escritura de varias líneas de comandos:

```
1Ø PRINT"PRIMERA LINEA"
2Ø A = 5
3Ø B = 1Ø
4Ø PRINT A;B;A + B
5Ø STOP
```

(Recuerde que al final de cada línea debe pulsar sin falta la tecla RETURN ).

Escriba ahora LIST para listar el programa y luego GOTO 1Ø para ejecutarlo. Compruebe si los resultados que aparecen en la pantalla son los que usted esperaba. La palabra STOP incorporada en un programa hace que se pare su ejecución al llegar a la misma y que aparezca el mensaje READY.

El orden correlativo de entrada de los comandos en la memoria se mantiene incluso si se escriben en otro orden. Por ejemplo, si usted escribe:

```
3Ø B = 1Ø
1Ø PRINT"PRIMERA LINEA"
4Ø PRINT A;B;A + B
5Ø STOP
2Ø A = 5
```

Estos comandos se entrarán, listarán y ejecutarán en el orden 10, 20, 30, 40, 50. Para comprobarlo, borre la memoria NEW y escriba la anterior lista de comandos, u otra lista no consecutiva.

Numere siempre las líneas de un programa en incrementos de 1Ø. De este modo usted no tiene problema para intercalar nuevas líneas de comandos; use simplemente números intermedios, como 15 ó 38 por ejemplo.

La razón de entrar los comandos en la memoria para su ulterior ejecución, en vez de mecanografiarlos y ejecutarlos directamente, es doble:

- Los comandos ya almacenados en la memoria se ejecutan mucho más rápidamente que si se tuvieran que escribir cada vez de nuevo en el teclado;

- Los comandos cargados en la memoria pueden ejecutarse tantas veces como se quiera. En la práctica, casi todas las tareas a ejecutar por ordenadores incluyen partes de programa de tipo repetitivo, que al entrarse en la memoria han de mecanografiarse una sola vez.

La forma más fácil de obtener la ejecución repetida de una parte del programa es con el comando GOTO, seguido del número de línea al que se desea reanudar la ejecución. Considere el siguiente programa:

```
10 PRINT"NORTE"
20 PRINT"OESTE"
30 PRINT"SUR"
40 PRINT"ESTE"
50 GOTO 10
```

El programa empezará su ejecución en la línea 1Ø, y al llegar a la 5Ø volverá a la n.º 1Ø. De este modo el programa es ejecutado continuamente hasta que usted lo para con RUN STOP o desconecta el ordenador.

Para empezar su ejecución, escriba GOTO 1Ø y pulse la tecla RETURN.

Vd. verá que el programa es ejecutado demasiado deprisa para poderlo leer. Para desacelerar su ejecución pulse y mantenga apretada la tecla CTRL (ensáyelo). Para parar el programa, pulse RUN STOP.

A continuación se expone un ejemplo práctico que ilustra la ventaja de numerar las líneas de programas en incrementos de 1Ø.

Supongamos que usted quiere intercalar direcciones intermedias entre los puntos cardinales del anterior programa, tales como NOROESTE, SUROESTE, etc.

Usted necesita nuevas instrucciones, que debe intercalar entre las ya existentes. Las puede, pues, numerar: 15, 25, 35 y 45.

Escriba:

```
15 PRINT"NOROESTE"
25 PRINT"SUROESTE"
35 PRINT"SURESTE"
45 PRINT"NORDESTE"
```

Para comprobar si las líneas se han intercalado correctamente, liste el nuevo programa con LIST: verá que las nuevas líneas se han introducido entre las anteriores. Ejecute el programa y verá también que la nueva secuencia de las instrucciones es ejecutada correctamente.

Para practicar, escriba sus propios programas y ensaye su ejecución. Utilice idénticos números de línea. Si usa signos gráficos en vez de palabras, obtendrá interesantes figuras en la pantalla.

En vez del comando GOTO 1Ø usado hasta aquí para empezar la ejecución de un programa, hay otro comando más práctico: RUN. Usted escribe simplemente RUN y pulsa la tecla RETURN y el programa empieza a ejecutarse automáticamente en su primera línea.

Cuando usted pone punto y coma después de una cadena, el C-64 no inicia una nueva línea al ejecutar el programa (sin embargo, al final de línea debe seguir pulsando la tecla RETURN, como en una línea normal). El C-64 empezará en tal caso nueva línea sólo cuando el mensaje en la pantalla no quepa en una misma línea. Con el uso del punto y coma, un programa sencillo, como el listado a continuación, permite llenar rápidamente toda la pantalla con curiosos signos. Ensáyelo y explique porqué se producen estas series interminables de signos gráficos.

```
10 PRINT "├─┤ ┬┴ ┌─┐ └ ";
20 GOTO 10
```

*Fin del Experimento 5.1*

# EXPERIMENTO 5·2



Cuando en un programa se incluye una secuencia de comandos que se ejecuta repetidamente, la misma recibe el nombre de "bucle". Un bucle puede estar formado por gran número de diferentes comandos, incluyendo una instrucción LET que dé un nuevo valor a una variable. Estudie el siguiente bucle de programa y trate de predecir su acción.

```
1Ø LET A = 1
2Ø PRINT A
3Ø LET A = A + 1
4Ø GOTO 2Ø
```

Imagine que usted es el ordenador y que tiene que hacer lo que se indica en cada línea. Escriba en el siguiente recuadro lo que ocurre a la variable A y al valor que se le asigna. No continúe la lectura de la presente lección hasta haber reflexionado y escrito con lápiz las soluciones en el recuadro.

(y así sucesivamente)

Ahora entre este programa en la memoria del ordenador y ejecútelo, manteniendo la tecla CTRL apretada para desacelerarlo (pero no apriete CTRL hasta después de haber escrito RUN y pulsado RETURN).

Sin duda usted habrá encontrado ese problema muy fácil, pero no obstante a continuación se explican las sucesivas fases de su ejecución.

El programa empieza realizando el comando etiquetado 1Ø, es decir asigna el valor 1 a la variable A. El siguiente comando exhibe este valor en la pantalla.

En la línea 3Ø A es sustituida por A + 1, por lo que el resultado es 2. En la siguiente, el comando GOTO restituye la ejecución a la línea 2Ø; al mostrarse de nuevo el valor de A, se ha convertido en 2. La secuencia o bucle constituido por las líneas 2Ø, 3Ø y 4Ø se repite indefinidamente, cada vez incrementando el valor A en 1.

Para adquirir práctica en esta clase de programación, estudie los dos bucles siguientes y trate de predecir los resultados de su ejecución. Escriba esos resultados debajo de cada secuencia de programa. (Recuerde que el signo ★ significa "multiplicado por").

```
1Ø B = Ø
2Ø PRINT B
3Ø B = B + 3
4Ø GOTO 2Ø
```

```
1Ø A = 1
2Ø B = A★A
3Ø PRINT A,B
4Ø A = A + 1
5Ø GOTO 2Ø
```

Entre esos dos programas y ejecútelos para comprobar si usted adivinó las respuestas.

Usted puede usar este tipo de bucles también con cadenas. Ensaye el siguiente programa:

```
1Ø X$ = "★"
2Ø PRINT X$
3Ø X$ = X$ + "#"
4Ø GOTO 2Ø
```

Los sucesivos valores de la variable X$ a medida que se repita la ejecución del bucle serán: ★, ★#, ★##, ★###, etc. La cadena X$ aparece en pantalla cada vez más larga. Al cabo de unos 4Ø segundos, la cadena es tan larga que no cabe en la memoria de la máquina (el máximo número de caracteres permitidos es 255); por consiguiente, aparece un mensaje de error:

?STRING TOO LONG

ERROR IN 3Ø

La parte del mensaje "ERROR IN 3Ø" significa que el comando que no consiguió almacenar la cadena excesiva fue el etiquetado con el nº 3Ø.

He aquí otros dos programas que usted debe intentar descifrar:

```
1Ø A$ = "+ +"
2Ø PRINT A$
3Ø A$ = "A" + A$ + "—"
4Ø GOTO 2Ø
```

```
1Ø A$ = "XY"
2Ø PRINT A$
3Ø A$ = A$ + A$
4Ø GOTO 2Ø
```

Recuerde que si una letra está dentro de una cadena, es decir, está entrecomillada, es simplemente una letra y no una variable. Por ejemplo, "X" no tiene nada que ver con X ni con X$.

Como último ejercicio, escriba un programa con un bucle sencillo y ejecútelo durante 1 minuto, cronometrando con el reloj. Entonces pare el programa y cuente las veces que se ha ejecutado el bucle. Luego calcule el número de ejecuciones de los comandos por segundo y anótelo en lápiz aquí:

*Fin del Experimento 5.2*

El programa de autoevaluación de esta unidad de estudio lleva el título UNI5CUEST.

# UNIDAD: 6


| | | |
|---|---|---|
| EXPERIMENTO 6.1 | PAG. | 37 |
| EXPERIMENTO 6.2 | | 38 |
| EXPERIMENTO 6.3 | | 40 |
| EXPERIMENTO 6.4 | | 42 |

El objetivo de este curso es enseñarle a desarrollar y construir sus propios programas. A este fin, en la presente unidad de estudio se exponen varias técnicas y procedimientos para comprobar, corregir y modificar sus programas. Estudie, pues, esta lección con detenimiento, ya que las técnicas en ella expuestas deberán ser aplicadas a menudo en el desarrollo del curso.



# EXPERIMENTO 6·1

Cargue en la memoria del ordenador y ejecute el programa de esta unidad de estudio titulado FRASES. Observe las frases que aparecen en la pantalla. Son frases construidas al azar, a partir de una breve lista de alternativas. Por el momento no nos preocupemos de cómo funciona este programa (su principio es en realidad muy simple); usémoslo simplemente para aprender como listar, modificar y conservar programas largos.

Cuando haya observado adecuadamente las frases, pare el programa con la tecla RUN/STOP y lístelo LIST. Pronto verá que es demasiado largo para visualizarlo en la pantalla, y que a medida que van apareciendo nuevas líneas en la parte inferior de la pantalla, van desapareciendo otras por su parte superior. Al final quedan visibles sólo los últimos comandos.

Para solventar esta anomalía, LIST puede usarse en otras versiones, cinco en total. Ensaye cada versión a medida que la lea a continuación.

- Usted puede listar todo el programa con sólo LIST; si no cabe en la pantalla, use alguna de las siguientes variantes.

- Es posible listar una sola línea especificando su número; por ejemplo:

LIST 11ØØ

- Es posible listar todos los comandos hasta un número de línea dado anteponiéndose un guión, como por ejemplo

LIST —8Ø

- Es posible listar todos los comandos a partir de un número de línea dado escribiendo un guión después de este:

LIST 9Ø9Ø–

- Finalmente, es posible listar todos los comandos comprendidos entre dos números de línea especificándolos de la siguiente forma:

LIST 2ØØØ—2Ø9Ø

Use ahora esas diferentes versiones del comando LIST para poder ver diversas partes del programa. Comprobará que los números de línea no van siempre en incrementos de 1Ø; la razón de ello es que el programa se modificó en varias ocasiones, intercalando nuevas líneas.

Al principio del programa y a diferentes intervalos verá el comando REM, seguido de una observación en castellano. REM es la forma abreviada de "remark", y se usa para introducir notas que faciliten la comprensión del programa, aún cuando no afectan su ejecución. Cuando empiece a escribir programas, introduzca frecuentes comandos REM haciendo referencia a aspectos difíciles de interpretar.

Cuando esté seguro de haber comprendido plenamente la función de las diferentes versiones del comando LIST, úselas en aplicaciones como las siguientes:

(a) Use LIST con un número de línea que no esté en el programa.

(ensaye LIST 65Ø)

(b) Especifique una gama de números de línea erróneos.

(ensaye LIST 11ØØ—1ØØØ)

(c) Use LIST incorporado en un programa; para ello, escriba NEW, mecanografíe el siguiente programa y ejecútelo:

```
1Ø PRINT"LISTA PRECIOS"
2Ø LIST
3Ø GOTO 1Ø
```

*Fin del Experimento 6.1*

# EXPERIMENTO 6·2



A continuación se explica como modificar programas. Estas modificaciones pueden ser de tres tipos:

(a) Supresión de comandos existentes.

(b) Adición o sustitución de comandos existentes.

(c) Corrección o sustitución de comandos existentes.

## Supresión de Comandos Existentes.

Hay cinco formas de cancelar un comando incorporado en un programa que está cargado en la memoria del ordenador, pero tres de ellas exigen el borrado o cambio de todo el programa, por lo que su efecto es drástico.

Para borrar todo el programa:

- desconectar el C-64
- escribir el comando NEW
- cargar un nuevo programa desde un cassette o disco.

Para borrar un sólo comando:

- escribir tan sólo su número de línea
- escribir otro comando con el mismo número de línea.

Vuelva a cargar en la memoria del C-64 el programa titulado FRASES y borre unas cuantas líneas (seleccione las que contienen la palabra clave REM). Para estar seguro de que las líneas se han suprimido, liste las correspondientes partes del programa antes y después de la operación de borrado.

## Adición de Nuevos Comandos.

Pueden incorporarse nuevas líneas de programa escribiéndolas con un número de línea adecuado. El comando se insertará automáticamente en el lugar que corresponde según el número de línea asignado.

Usted ya practicó la inserción de comandos en la unidad de estudio 5, pero puede aprovechar esta oportunidad para insertar comandos

REM. Sin embargo, asegúrese de no sustituir ninguna instrucción existente, o de lo contrario el programa no funcionará.

## Modificación de Comandos Existentes.

La forma más drástica de modificar un comando consiste en volverlo a mecanografiar en la forma deseada, utilizando el mismo número de línea.

Ensaye una modificación de este tipo. Cambie la línea número 5, que contiene los derechos de autor, de modo que muestre el nombre de usted:



```
        LIST 5

        5 REM COPYRIGHT © ANDREW
          COLIN 1981
Usted
escribe: 5 REM JORGE RUIZ

        LIST 5

        5 REM JORGE RUIZ
```

Practique modificando otras líneas, pero seleccionando siempre las que contengan instrucciones REM, de lo contrario lo más probable que altere el programa y no pueda ejecutarse; modificaciones al azar suelen hacer los programas inservibles.

Si se desea introducir sólo un pequeño cambio, es a veces más fácil modificar el original (previamente listado en la pantalla) que mecanografiar una nueva línea. Ello se realiza con el cursor y a veces con la tecla INST DEL. Al terminar los cambios, pulsar la tecla RETURN, con lo que se consumará la modificación.

Supongamos que usted quiera modificar la línea 1ØØ del programa FRASES de modo que la misma se convierta en:

1ØØ REM GENERADOR ESTUPIDO DE FRASES

Liste el comando 1ØØ, ponga el cursor después de la R de GENERADOR e inserte 9 espacios en blanco (usando las teclas SHIFT y INST DEL). Luego mecanografie la palabra ESTUPIDO, observe que todo es correcto y pulse RETURN. Liste de nuevo la línea 100 para asegurarse de que ha adoptado la nueva forma.

Ensaye unas cuantas modificaciones más de este tipo, seleccionando siempre comandos REM. Recuerde que si después del cambio no pulsa la tecla RETURN la máquina no consumará el cambio.

Finalmente, ejecute el programa. Si no funciona, es indicio de que usted ha cometido algún error en la edición como por ejemplo borrando o modificando una instrucción. Si ha sido así, no se preocupe, es cosa frecuente; simplemente vuelva a cargar el programa desde el cassette.

Usted habrá observado que el programa FRASES emite opiniones sobre conocidas personalidades. Las listas de alternativas son muy limitadas: están contenidas en los comandos 9Ø7Ø (para hombres) y 91ØØ (para mujeres). En la parte final de este experimento usted puede modificar estas listas de modo que el programa construya frases sobre su familia y amigos.

Tanto el comando 9Ø7Ø como 91ØØ tienen la palabra clave DATA, seguida de una lista de nombres, separados por comas. El último nombre va seguido de una coma y la letra Z*.

En cada comando pueden incluirse tantos nombres como se quiera, aunque si ocupan más de 2 líneas de pantalla deberá usarse un segundo comando DATA (con un número de línea que represente un incremento); puede usarse incluso un tercer o cuarto comando, pero es esencial que el último comando DATA de cada grupo termine con Z.

Por ejemplo, usted podría cambiar los nombres de las líneas 9Ø7Ø y 91ØØ de la siguiente forma:

```
9Ø7ØDATAJOSE,JORGE,
ANTONIO,SR.VILA,MARIO,
DR.PRESAS,ELMEDICO,Z
91ØØDATAABUELA,SUSANA
VIOLETA, SRA, CASTILLO,
DRA.ISERN,PRIMA,MARIA
91ØDATAMONTSE,NURIA,
TIA ENCARNA,ROSALIA,Z
```

Cuando haya efectuado esos cambios, ejecute de nuevo el programa. Si lo que aparece en la pantalla no tiene sentido, asegúrese de que haya puesto una coma entre cada nombre y de que el último nombre va seguido de la Z.

Cuando sepa realizar este tipo de modificaciones, cambie otras listas de palabras del programa. Se hallan en las siguientes línea:

9ØØ Acciones que la gente hace por sí misma (verbos intransitivos)

9Ø1Ø Acciones que recaen en otras personas (verbos transitivos)

9Ø2Ø Acciones con prendas de vestir (verbos transitivos)

9Ø3Ø Prendas masculinas

9Ø4Ø Prendas femeninas

---

* El uso de la letra Z al final de un comando DATA es una característica de este programa, no del lenguaje BASIC en general. La mayoría de los programas utiliza comandos DATA sin necesidad de terminar con una Z.

9050 Una lista de adverbios y frases adverbiales describiendo acciones transitivas

9060 Una lista de adverbios y frases adverbiales describiendo acciones intransitivas

9070 Nombres masculinos

9080 Diferentes clases de hombres

9100 Nombres femeninos

9110 Adjetivos descriptivos de mujeres

9120 Diferentes clases de mujeres

Modifique esas listas a su discreción.

*Fin del Experimento 6.2*

# EXPERIMENTO 6·3



Cuando usted modificó el programa FRASES de modo que expresara originales opiniones sobre sus amigos, pensó quizás que la nueva versión se prestaría a ser ejecutada en reuniones o fiestas que dé en su casa. En realidad usted puede conservar esta variante del programa transfiriéndola a una cinta cassette, de la forma explicada a continuación.

Coja una cinta virgen o que no contenga ningún programa que quiera conservar. Ha de ser de buena calidad y lo más corta posible, pues se necesita sólo aproximadamente 1 minuto de grabación. Para estas aplicaciones son ideales las cintas cassette de Commodore.

Coloque esta nueva cinta en el cassette C-64 en lugar de la cinta FRASES utilizada hasta ahora en esta unidad de estudio, y rebobínela. Suelte todas las teclas del cassette. Pare la ejecución del programa FRASES, y escriba en el teclado:

SAVE"FAMILIA" y pulse la tecla **RETURN**

(en vez de FAMILIA puede usar cualquier otro nombre con que quiera designar la nueva versión de FRASES).

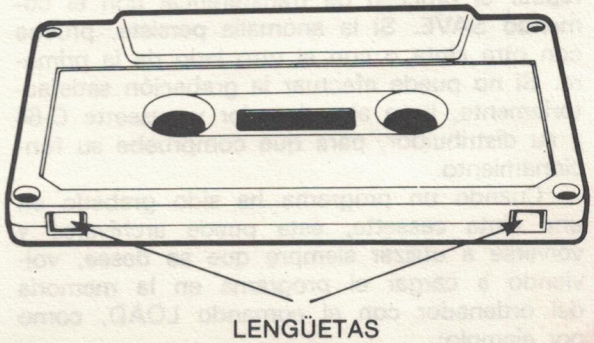
El ordenador responderá con el siguiente mensaje:

PRESS RECORD AND PLAY ON TAPE

(Pulse las teclas RECORD y PLAY del cassette).

Pulse, pues, ambas teclas simultáneamente.

Si la tecla RECORD no puede pulsarse, compruebe que en el cassette de la cinta no falten las lengüetas de "permiso de grabación", situadas en la parte posterior del cassette, como se indica en la figura.



LENGÜETAS

Si están retiradas no es posible grabar en la cinta. Su finalidad es proteger grabaciones importantes contra destrucción accidental. Si este fuera el caso, busque otra cinta.

Si todo marcha bien, la máquina exhibirá el mensaje:

SAVING FAMILIA
(Grabando FAMILIA)

y un momento después informará que la grabación está determinada con el mensaje:

READY

Aunque el programa ya esté grabado, conviene asegurarse. Puede haber surgido alguna anomalía: usted puede haberse olvidado de rebobinar la cinta o de pulsar el botón RECORD o la cinta puede tener un defecto que impida copiar una parte del programa. Teóricamente estas anomalías no deberían ocurrir, pero en la práctica suceden.



Para verificar si la grabación es correcta, rebobine la cinta y luego escriba en el teclado:

VERIFY "FAMILIA" RETURN

Al entrar el comando VERIFY (Verificar), en la pantalla aparecerá el mensaje:

PRESS PLAY ON TAPE
(Pulse la tecla PLAY del cassette)

Pulse, pues, la tecla PLAY (pero esta vez no la tecla RECORD). La máquina buscará el programa en la cinta y lo comparará con el existente en la memoria (naturalmente, usted no debe introducir ninguna modificación en el programa entre su grabación y el proceso de verificación).

Si todo está bien, en la pantalla aparecerá:

```
VERIFY"FAMILIA"
PRESS PLAY ON TAPE
OK
SEARCHING FOR FAMILIA
FOUND FAMILIA
VERIFYING
OK
```

Si la máquina encuentra un error o no llega al mensaje "FOUND FAMILIA", usted debe repetir el proceso de transferencia con el comando SAVE. Si la anomalía persiste, pruebe con otra cinta o con el otro lado de la primera. Si no puede efectuar la grabación satisfactoriamente, lleve el ordenador y cassette C-64 a su distribuidor, para que compruebe su funcionamiento.

Cuando un programa ha sido grabado en una cinta cassette, ésta puede archivarse y volverse a utilizar siempre que se desee, volviendo a cargar el programa en la memoria del ordenador con el comando LOAD, como por ejemplo:

LOAD"FAMILIA"

No es necesario esperar a que el programa se haya depurado a la perfección para grabarlo. Si usted está escribiendo un programa largo de su propia creación (o incluso cuando lo copia de un libro), vale la pena de grabarlo a intervalos regulares, como por ejemplo cada media hora. La razón de ello es que la memoria del C-64 no es tan fiable como una cinta; el ordenador no se averiará, pero pueden presentarse accidentes: se han dado casos de que una tormenta haya perturbado los datos contenidos en la memoria de un ordenador, puede producirse un corte de corriente o alguien de la casa puede tropezar con el cable tendido en el suelo y desenchufar el ordenador. Si por una razón de éstas usted pierde seis horas de trabajo, le sabrá mal no haber realizado grabaciones periódicas, ya que en este caso en la última grabación se halla disponible la mayor parte del trabajo y la pérdida será mínima.

Para mayor seguridad, conviene grabar utilizando dos cintas, que usa alternativamente. De este modo, aún en el caso de que la máquina se parara durante una grabación, con la mitad de la versión antigua borrada por la mitad de la nueva versión, le quedaría íntegra la cinta anterior.

Si produce una cinta de programa C-64 en un cassette normal sólo para sonido, oirá curiosos efectos sonoros.

*Fin del Experimento 6.3*

# EXPERIMENTO 6·4

En esta sección señalamos una sutil trampa en que pueden caer fácilmente los programadores del C-64, e indicamos la forma de salir de ella.

Empezamos, pues, preparándole la trampa: escriba el comando NEW para borrar la memoria y escriba el siguiente programa, insertando exactamente los espacios indicados y mirando en la pantalla mientras escribe:

```
10 PRINT "ESTA ES UN
A PRUEBA CON TRAMPA"
20 GOTO 10
```

Cuando haya introducido estas dos líneas, escriba RUN y pulse RETURN. Evidentemente, usted esperará que en la pantalla aparezca el mensaje siguiente, repetidamente:

"ESTA ES UNA PRUEBA CON TRAMPA"

Es muy posible que sea esto lo que vea en la pantalla. Claro está, si no ha caído en la trampa; pero es igualmente probable que lo que muestre la pantalla sea el siguiente mensaje de error:

```
"ESTA ES UNA PRUEBA CON TRAMPA" 20
?SYNTAX
ERROR IN 10
READY
```

Tanto si ha cometido el error como si no, siga leyendo, para averiguar porque no ha caído en la trampa o como salir de ella.

La razón del error (de haber incurrido en él) no es en modo alguno obvia. Si usted muestra este programa a cualquier experto en programación en BASIC no encontrará nada anómalo.

La dificultad reside en que la anchura de la pantalla es 40 caracteres. Dentro de la máquina, los comandos BASIC pueden alcanzar una longitud de hasta 80 caracteres, o sea 2 líneas de pantalla.

Al llegar al final de línea al mecanografiar un comando, el cursor pasa automáticamente al principio de la línea siguiente, para que usted siga escribiendo el resto del comando.

Para terminar el comando aún cuando el cursor se haya situado al principio de la línea siguiente, es necesario pulsar la tecla RETURN. Si, al ver la situación del cursor, se olvidó de pulsar esa tecla y prosiguió con la escritura del segundo comando, al pulsar finalmente RETURN todo lo escrito fue entrado como una sola instrucción.

Por su sintaxis, un comando de este tipo es erróneo y cuando se intenta ejecutarlo en la pantalla aparece el mensaje de error descrito.

Este tipo de error es especialmente difícil de detectar a menos que usted sepa lo que busca. La razón de ello es que si usted lista el programa las dos líneas aparecen separadas, tal como usted las escribió.

Hay, claro está, una forma para encontrar este error tan plausible, en el que puede caer todo programador: en vez de listar todo el programa o la sección en que se supone hay el error, liste sólo la línea señalada en el mensaje de error, en nuestro caso escriba: LIST 10. En la pantalla aparecerá no sólo la línea 10, sino la 20, poniéndose de manifiesto que las instrucciones no se separaron con la tecla RETURN. Para corregir la anomalía, vuelva a escribir ambos comandos totalmente, pulsando al final de cada uno la tecla RETURN

De todo ello, podemos extraer las dos enseñanzas siguientes:

(a) Termine siempre cada comando con RETURN, cualquiera que sea la posición del cursor.

(b) Cuando el C-64 muestre un mensaje de error en un comando y no vea en el mismo nada anómalo, lístelo separadamente y observe si incluye el segundo comando.

*Fin del Experimento 6.4*

# UNIDAD: 7


| | | |
|---|---|---|
| EXPERIMENTO 7.1 | PAG. | 46 |
| EXPERIMENTO 7.2 | | 47 |
| EXPERIMENTO 7.3 | | 50 |

Los programas que se escribieron en la Unidad 5 estaban exentos de todo control y tendían a continuar indefinidamente, a menos que se adoptara una medida drástica. En la presente unidad de estudio se explica cómo controlar los programas y conseguir que se paren cuando han cumplido su cometido.

Los temas expuestos a continuación son de importancia fundamental en la tecnología de ordenadores. Cuando usted los domine habrá realizado un importante avance en el camino de convertirse en programador. Lea las siguientes páginas lenta y detenidamente, y si hay algún punto del que no está seguro vuelva a estudiarlo. Esta perfecta comprensión de las ideas expuestas en esta lección es una condición para el satisfactorio progreso en las ulteriores unidades de estudio.



El control de los programas se basa en un concepto clave: la condición o cláusula condicional.

Por condiciones se entienden frases o suposiciones introducidas por la conjunción "si", seguidas de una consecuencia (en las siguientes frases, la condición va subrayada):

**"Si el tren ya ha salido**, tendrás que pasar la noche en un hotel"

**"Si el programa no opera,** busque el defecto y corríjalo"

Estas frases no son simples afirmaciones positivas o negativas, sino posibles supuestos, de cuya materialización se deducen determinadas consecuencias.

En el lenguaje BASIC, la conjunción "si" adopta la forma "IF", su equivalente inglés. IF es la palabra clave del comando y va seguida de los elementos corrientes en programas: variables numéricas, variables de cadenas, cantidades y cadenas. Las condiciones pueden cumplirse o no, es decir son reales o falsas en cada caso concreto, y se expresan utilizando seis relaciones posibles, que adoptan la forma de signos aritméticos. Por ejemplo,

A < 5

(donde < es un signo que significa "menor que"). Esa condición es verdadera si el valor de la variable A es "menor que" 5 (como por ejemplo 0,3 ó 4,98). Es falsa si A es igual o mayor que 5.

Otro ejemplo, utilizando cadenas:

N$ <> "JAIME"

(donde <> significa "diferente de")

Esta última condición es verdadera si la variable N$ tiene un valor que no sea "JAIME"; o sea será verdadera si N$ = "PEDRO" o N$ = "LUIS". Es falsa sólo en el caso de que N$ sea en realidad "JAIME".

Las seis relaciones usadas en BASIC son:

| | |
|---|---|
| = | (igual a) |
| < | (menor que) |
| > | (mayor que) |
| <> | (diferente de) |
| <= | (no mayor que) |
| >= | (no menor que) |

Las relaciones <>, <= y >= deben escribirse pulsando las dos respectivas teclas. Es posible que estos símbolos sean más conocidos a usted en las versiones ≠, ≤ y ≥, respectivamente, pero los creadores del lenguaje BASIC seleccionaron las formas indicadas por el hecho de que la mayoría de los ordenadores no poseen esas teclas.

Las citadas relaciones pueden usarse para constituir la condición entre pares de números o pares de cadenas; además, tanto los números como las cadenas pueden estar representados por variables. Así,

5>4 es verdadera, porque 5 es mayor que 4

7<6 es falsa, porque 7 es mayor que 6

si A = 10 y B = 7,

A>=B es verdadera, al igual que B<=7

Cuando las relaciones se usan entre cadenas, su evaluación se refiere al orden alfabético, o sea que

"PERRO" > "GATO" es verdadera,

mientra que "JAIME" > "JIMMY" es falsa.

o

# EXPERIMENTO 7·1

Supongamos que el ordenador ha realizado las siguientes tres instrucciones:

```
LET A$ = "JOAN"
LET X = 5
LET Y = 7
```

Calcule si las condiciones de la siguiente tabla son verdaderas o falsas e indíquelo en la correspondiente columna:

| Condición | Valor (falsa o verdadera) |
|---|---|
| X < 7 | |
| X >= 5 | |
| A$<>"X" | |
| Y <> X | |
| A$ < "FRANCES" | |
| A$ > "JOAN" | |
| Y = 8 | |

Las cantidades en cada lado de la relación pueden ser expresiones, como con LET. Las expresiones pueden ser tan complejas como usted quiera, pero ambos términos de la condición deben ser del mismo tipo: si una condición tuviera un número en un lado y una cadena en el otro, el C-64 se pararía y exhibiría un mensaje de error.

Suponga ahora que A$, X y Y tienen el mismo valor que en la anterior tabla, y, sobre dicha base, evalúe las condiciones siguientes:

| Condición | Valor (falso o verdadero) |
|---|---|
| A$+"NE"<>"JOANNE" | |
| 5 > X | |
| X + Y <> 13 | |
| X + 2 = Y | |

Compruebe si sus respuestas han sido correctas, consultando el Apéndice B, en que se relacionan las respuestas.

*Fin del Experimento 7.1*

# EXPERIMENTO 7·2



El elemento principal de control en el lenguaje BASIC es el comando IF. Está integrado por la palabra clave IF seguida de una condición, la palabra THEN y un número de línea. Su acción es muy similar a la de GOTO, pero se diferencia en que el salto tiene lugar sólo si se cumple el requisito estipulado en la condición.

He aquí un ejemplo de comando IF:

```
IF X$ <> "ABBBB" THEN 2Ø
```

En este caso la condición es X$ <> "ABBBB", y la formulación del comando indica a la máquina que debe saltar a 2Ø si X$ es diferente de "ABBBB". Si esta condición es falsa, la máquina continuará ejecutando los siguientes comandos, por orden numérico ascendente.

Es posible que en un principio este comando le parezca algo absurdo; usted aducirá: ¿X$ es diferente o no de ABBBB según lo que se haya puesto delante, o sea que el programador sabía de antemano la respuesta.

Esta opinión parece lógica y plausible, pero no refleja la realidad, por dos razones fundamentales:

- Suponga que el comando IF sea un bucle en que alguna variable cambie de valor en cada ciclo de ejecución; en tal caso, la condición será verdadera en unos casos y falsa en otros.
- Suponga que usted escribe un programa para que lo use otra persona; en tal caso, usted no puede prever las acciones del usuario y el programa dependerá de tales acciones. Un ejemplo ilustrativo de esta dependencia son los programas de este curso, en que el autor no tenía idea de lo que usted iba a decir en cada caso, pero desarrolló los programas quizá con sentido común.

La inclusión de una instrucción IF en un bucle es una buena forma de parar la ejecución del programa al cabo de un número determinado de ciclos de funcionamiento. Mecanografíe el siguiente programa y ejecútelo:

```
1Ø X$ = "A"
2Ø PRINT X$
3Ø X$ = X$ + "B"
4Ø GOTO 2Ø
5Ø STOP
```

Este programa funciona indefinidamente, llenando la pantalla con cadenas cada vez más largas de B, hasta que se agota la memoria. La instrucción STOP de la línea 5Ø nunca se alcanza.

Ahora pare su ejecución y cambie la línea 4Ø por la siguiente instrucción:

```
4Ø IF X$ <> "ABBBB" THEN 2Ø
```

Cuando usted ejecute el programa, en la pantalla aparecerá:

```
A
AB
ABB
ABBB
```

y se para.

La razón de este paro reside en la condición X$ <> "ABBBB" . Cuando el programa ejecuta repetidamente el bucle, la condición es primero verdadera (porque X$ es AB, ABB y ABBB, todas ellas diferentes de ABBBB); por consiguiente, la instrucción IF se comporta como un simple comando GOTO 2Ø. Cuando el valor de la variable X$ llèga a ABBBB, la condición es falsa, por lo que el programa no salta a la línea indicada, sino que prosigue hasta el final, donde se para al encontrar el comando STOP.

Para ensayar la construcción de condiciones, modifique de diferentes modos la instrucción IF del anterior bucle y observe el efecto cuando ejecute el programa. En todas las condiciones seleccionadas asegúrese de que finalmente es falsa, ya que de lo contrario el programa no se detendrá.

Pruebe, por ejemplo, las siguientes versiones de la condición incorporada en la instrucción IF:

```
X$<>"AB"
X$<>"ABBBBBBBBBBB"
X$<>"ABBA"
```

La misma técnica de control puede aplicarse a variables numéricas. Escriba en el teclado:

```
1Ø P = Ø
2Ø PRINT P,P★P
3Ø P=P+1
4Ø GOTO 2Ø
5Ø STOP
```

Ejecute este programa, observe su acción, párelo y cambie la línea 40 a la siguiente versión:

```
4Ø IF P<11 THEN 2Ø
```

Ahora vuelve a ejecutarse el programa: muestra dos columnas de números de aspecto familiar, que podrían ser útiles para quien no conociera de memoria el cuadrado de los números. Como programa tiene una sola falta: las dos columnas no llevan título y por consiguiente su significado no es obvio a primera vista.

Podemos solventar esta deficiencia asignando un título o definición a cada columna, como por ejemplo:

| NUM. | CUADRADO |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 4 |
| 3 | 9 |
| .. | .... |
| etc. | |

Los títulos deben exhibirse por encima de la primera línea de números, por lo que también su comando debe estar antepuesto al resto de instrucciones. Como ya se usaba el número de línea 1Ø, podemos usar por ejemplo el número 5. El comando a usar será naturalmente PRINT, con las dos cadenas: "NUM", y "CUADRADO". Al poner una coma entre ambas el espaciado será idéntico al de las cantidades. El programa tiene ahora la siguiente forma:

```
5 PRINT "NUM.","CUADRADO"
1Ø P=Ø
2Ø PRINT P,P★P
3Ø P=P+1
4Ø IF P<11 THEN 2Ø
5Ø STOP
```

Ejecute este programa y examine la pantalla.

Entre el título y la primera línea de números puede poner una línea en blanco, si lo desea. Ello se consigue intercalando un comando PRINT sólo (que no vaya seguido de ningún valor ni cadena). Para ensayarlo, escriba:

```
7 PRINT
```

A continuación le pediremos que empiece a escribir sus propios programas, pero antes examinemos los dos programs que acabamos de explicar, a fin de extraer algunas conclusiones de aplicación general. Esos programas eran:

(1)

```
1Ø X$="A"
2Ø PRINT X$
3Ø X$=X$+"B"
4Ø IF X$<>"ABBBB" THEN 2Ø
5Ø STOP
```

(2)

```
5 PRINT"NUM.","CUADRADO"
7 PRINT
1Ø P=Ø
2Ø PRINT P,P★P
3Ø P=P+1
4Ø IF P<11 THEN 2Ø
5Ø STOP
```

Si prescindimos de los comandos para la inclusión de los títulos (5 y 7) en el segundo programa, ambas secuencias siguen una pauta similar:

1. Hay una variable que cambia cada vez que se repite el bucle: X$ en el primer programa y P en el segundo. Esta variable se suele denominar variable de control y puede ser un número o una cadena.

2. Hay un comando que asigna un valor inicial a la variable de control. Este comando es externo al bucle (es decir, se ejecuta una sola vez, en vez de repetirse con cada ciclo del bucle).

3. Hay un comando que se ejecuta con cada valor de la variable de control. En nuestros ejemplos, son:

   PRINT X$

   y PRINT P,P★P

   En la práctica, esta parte del bucle puede extenderse, de modo que incluya cualquier número de comandos, todos los cuales son ejecutados con cada valor de la variable de control. Este grupo se denomina "cuerpo" del bucle.

4. Hay un incremento, es decir la cantidad con que se incrementa la variable de control cada vez que se ejecuta el bucle. En nuestros ejemplos, X$ se incrementa añadiendo una B, y P se incrementa en una unidad. Los incrementos pueden tener cualquier valor; por ejemplo, una cadena podría incrementarse cada vez en 5 símbolos, y un número podría incrementarse en 2 o cualquier otro valor por cada ciclo. En vez de incremento puede seleccionarse un decremento, empezando con un valor

alto y reduciéndolo en el valor especificado cada vez que se ejecuta el bucle. Puede decirse, pues, que en cada bucle hay siempre un comando que incrementa o decrementa la variable de control con cada ciclo de ejecución.

5. El límite de la variable de control se llama "valor final", y al alcanzarse cesa la ejecución cíclica del bucle. Este valor final se especifica en el comando IF, expresando una condición que será "verdadera", si el bucle debe ejecutarse de nuevo o "falsa", si la variable de control ha alcanzado su valor final.

En la tabla siguiente se relacionan varios bucles de programa. Examínelo detenidamente y anote en las sucesivas columnas el nombre de la variable de control, el valor inicial, el valor final, el incremento (o decremento) y el número de veces o "ciclos" que se ejecuta. Para calcular este último parámetro a veces es útil anotar el valor de la variable de control repetidamente en un papel, para ver el número de ciclos requeridos para que se alcance el valor final.



| | Variable de control | Valor inicial | Valor final | Incremento | Nº de ciclos del bucle |
|---|---|---|---|---|---|
| 10 X$="A"<br>20 PRINT X$<br>30 X$=X$+"B"<br>40 IF X$<>"ABBBB"THEN 20<br>50 STOP | X$ | "A" | "ABBB" | "B" | 4 |
| 10 P=0<br>20 PRINT P,P*P<br>30 P=P+P<br>40 IF P < 11 THEN 20<br>50 STOP | P | 0 | 10 | +1 | 11 |
| 10 Y$="Z"<br>20 PRINT Y$<br>30 Y$=Y$+"XY"<br>40 IF Y$ <>"ZXYXYXY"THEN 20<br>50 STOP | | | | | |
| 10 R=5<br>20 PRINT R, R/8<br>30 R=R+3<br>40 IF R < 17 THEN 20<br>50 STOP | | | | | |
| 10 C=27<br>20 H=30-C<br>30 PRINT C,H<br>40 C=C-5<br>50 IF C > 2 THEN 20<br>60 STOP | | | | | |

Cuando haya completado la tabla, compare sus respuestas con las soluciones dadas en el Apéndice B.

*Fin del Experimento 7.2*

# EXPERIMENTO 7·3



Cuando usted construye un programa debe empezar trazando cierto plan básico, a partir del cual empezará a deducir las sucesivas líneas de instrucciones. Escriba los comandos primero sobre una hoja de papel, utilizando lápiz y goma de borrar. No trate de componer sus programas directamente en el teclado del ordenador, sería infinitamente más difícil y tedioso. Escribir un programa sin plan previo sería como querer edificar una vivienda de diez plantas sin trazar plano alguno; lo más probable es que empezara a derrumbarse antes de completarla.

Cuando usted diseña un bucle de programa debe decidir ante todo los componentes esenciales: el nombre de la variable de control, su tipo, el valor inicial y final, el incremento y los elementos que integran el cuerpo del bucle. Sólo después de decidir esas cuestiones puede combinarlas siguiendo la pauta normal del programa.

He aquí un ejemplo de planificación del trabajo:

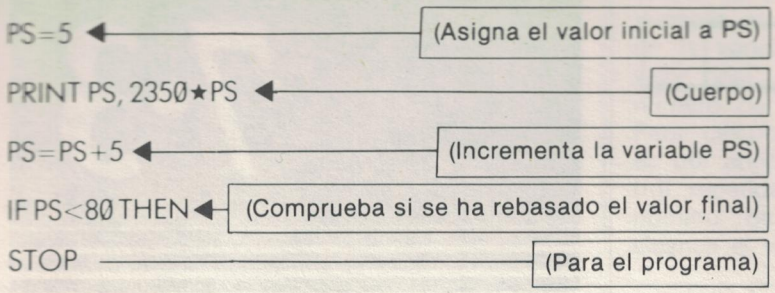
Supongamos que quiera convertir libras esterlinas a liras italianas, para componer una tabla de cambio. Imaginemos un cambio de 2.350 liras por libra esterlina y que deseamos una tabla equivalente en la gama de 5 a 75 libras, en intervalos de 5. Podemos empezar el programa de modo que aparezcan en la pantalla los siguientes títulos y cantidades:

| £ | LIRAS |
|---|---|
| 5 | 11750 |
| 10 | 23500 |
| ..... | |

etc.

Tratemos de componer primero el bucle. La variable de control será evidentemente un número y podemos designarla, por ejemplo, PS, lo que nos recuerda las iniciales de "pound sterling", o libra esterlina. El valor inicial será 5 y el final 75. El incremento, como hemos dicho, 5. El cuerpo del bucle exhibirá un valor en libras y en la siguiente columna su equivalente en liras, o sea 2.350 veces mayor.

Los elementos de nuestro bucle podrán construirse de la siguiente forma:






(No ponemos el número de línea después de THEN porque aún no hemos numerado nuestras instrucciones).

Antes de escribir el programa, decidamos el título de cada columna; por ejemplo:

```
PRINT"£","LIRAS"
```

volvamos a escribir PRINT

(para que quede una línea en blanco)

Ahora podemos combinar las partes en el siguiente programa:

```
10 PRINT "£", "LIRAS",
20 PRINT
30 PS=5
40 PRINT PS, 2350★PS
50 PS=PS+5
60 IF PS < 80 THEN 40
70 STOP
```

Recuerde, pues, esta regla básica: no trate de improvisar sus programas escribiéndolos directamente en el teclado. Si lo hace, no sólo perderá tiempo, sino que nunca llegará a ser un gran programador.

Ahora practique la construcción de bucles creando los siguientes programas:

1. Escriba un programa que muestre en la pantalla una pauta de estrellas en número creciente de la siguiente forma:

```
★
★★
★★★
★★★★
.....
hasta
★ ★ ★ ★ ★ ★ ★ ★ ★ ★
```



Dream commodore

El uso del signo "=" en BASIC no coincide exactamente con el que se le da en matemáticas, pudiendo crear confusiones a algunos seguidores de este curso; aunque no sea éste su caso, le será de utilidad leer a continuación el significado de "=" en BASIC.

En matemáticas, "=" se emplea en ecuaciones para indicar que dos expresiones diferentes tienen idéntico valor. Por ejemplo, si el profesor de aritmética escribe en la pizarra.

"2x + 5 = 9"

todo el mundo sabrá que el valor de X es 2.

En BASIC, "=" se usa en dos sentidos diferentes, ninguno de los cuales corresponde al que se le da en matemáticas.



Usado con un comando LET, el signo "=" significa "tendrá el valor de". Su efecto es hacer que el C-64 calcule el valor de la expresión escrita a la derecha del signo y lo asigne a la variable que consta a su izquierda. El significado normal del término inglés LET explica claramente esta función, sin embargo esta palabra clave de BASIC normalmente se omite, con lo que el comando adquiere la forma de una ecuación, aunque no lo sea en absoluto. Por ejemplo, en BASIC,

Y = X + 2

no informa al ordenador de que Y es igual a X + 2, sino que le pide que calcule el valor de la expresión X + 2 y asigne el resultado a la variable Y. Considere los siguientes casos:

- Q=Q+5 — No es un contrasentido matemático, sino una instrucción LET perfectamente válida.
- P=Q y Q=P — No son en modo alguno equivalentes, sino dos comandos distintos.
- X+1=5 — No es un comando válido en BASIC.

¿Ve usted en cada caso el significado de estas instrucciones BASIC y el porqué una forma incompatible en matemáticas puede ser perfectamente lógica en un comando LET?

El otro uso de "=" es en las condiciones del comando IF. Usted recordará que "=" es una de las seis posibles relaciones entre cantidades. Por ejemplo:

```
IF X=Y+2 THEN 100

IF N$="YES" THEN 150
```

En estas condiciones, no se presupone una igualdad, sino que se intenta averiguar si la misma existe; según haya igualdad o no se adoptará una acción u otra. En las condiciones con IF, el signo "=" tiene la misma fuerza lógica que cualquier otra relación, como < ó >. Por dicha razón, sería quizás preferibe en ocasiones no designar el signo "=" como "igual a", a menos que se sea consciente de la diferencia inherente al mismo con respecto al significado normal en matemáticas.

Resumiendo:

En el lenguaje BASIC, el signo "=" se usa con el comando LET con el significado de "que tenga el valor de", y en condiciones IF con el de "es lo mismo que", pero sin que tenga necesariamente que serlo, por ser sólo un supuesto.

El programa de autotest a usar con esta unidad de estudio se denomina UNI7CUEST.

# UNIDAD: 8


| | | |
|---|---|---|
| EXPERIMENTO 8.1 | PAG. | 60 |
| EXPERIMENTO 8.2 | | 61 |
| EXPERIMENTO 8.3 | | 63 |

Usted ha empezado ya a escribir breves programas, muy simplificados, y a medida que aumenten sus conocimientos, experiencia y habilidad irá componiéndolos de mayor longitud, complejidad e interés. Para ilustrar el grado de complejidad que pueden adquirir los programas, en la siguiente tabla se indica el número de instrucciones que integran algunos programas:

| Programa | Número de Comandos |
|---|---|
| Cambio de liras a £ (Unidad 7 | 7 |
| Programa UNI3CUEST | apróx. 1ØØ |
| Programa de juego de ajedrez | apróx. 5.ØØØ |
| Programa para controlar un robot en un proceso industrial | apróx. 25.ØØØ |
| Programa para un sistema de reserva de plazas en una compañía de aviación | apróx. 5.ØØØ.ØØØ |

Los programas que poseen más de 5.000 comandos son producidos normalmente por equipos de programadores, ya que de lo contrario exigirían demasiado tiempo. Sin embargo, el campo de acción para programadores individuales es muy extenso.

Cuando usted trabaje en programación, a menudo se encontrará con que un programa que ha escrito no es ejecutado por el ordenador o no da los resultados esperados. En la presente unidad de estudio se explican algunas de las formas con que usted puede evitar que ello ocurra o que le permitirán solventar tales dificultades. Lea detenidamente el texto y realice con igual esmero todos los ejercicios. Además, consulte esta unidad siempre que se encuentre con dificultades que no pueda superar.

Todo estudiante de programación llega un momento en que se encuentra frente a un problema aparentemente insalvable; sin embargo, cuando ello le ocurra a usted no se desanime, pensando que la programación no es para usted, ni juzgue su programa como 99% correcto y, considérelo como bueno, siga adelante. Restar importancia al problema es tan pernicioso como descorazonarse. Lo que debe hacer es buscar el error y corregirlo. No hay programador que no se encuentre ante esa dificultad, aún cuando haga 25 años que trabaje con ordenadores.

Los errores de programación pueden catalogarse en tres grupos generales:

El primero y más corriente es el que da lugar a un mensaje de error de sintaxis en la pantalla cuando se trata de ejecutar el comando erróneo. Dicho mensaje significa que en la instrucción se ha infringido alguna regla del lenguaje BASIC; puede ser un simple error de mecanografía o puede haber demasiadas comillas o ser un número insuficiente. La mayoría de los errores de sintaxis se encuentran con facilidad, si usted sabe dónde están. El propio mensaje da una idea de la anomalía; vea el Apéndice C, en que se da una lista de comprobación del tipo de error en cuestión.

El segundo tipo de error en programas que no operan es la imposibilidad de ejecutar un comando dado. Supongamos que la máquina llega a la instrucción

```
13Ø GOTO 5ØØ
```

y que no hay ninguna línea del programa que tenga el número 500. En tales casos la máquina se parará y exhibirá el siguiente mensaje:

```
UNDEFINED STATEMENT IN 13Ø
```

es decir "instrucción no definida en 130"; o sea que no sólo los mensaje de error son en inglés, sino que es una especie de argot de la informática, difícil de interpretar para el principiante; cuando no entienda un mensaje de error, consulte el Apéndice C, hasta que esté familiarizado con su significado.

La tercera clase de defectos en los programas que usted escribe es más difícil de localizar y corregir: no se lee ningún mensaje de error, el ordenador simplemente muestra una respuesta errónea o se atasca en un bucle, sin visualizar nada. En tales casos, interrumpa la ejecución, liste el programa y examínelo detenidamente. En la mayoría de las veces bastará un cuidadoso examen para determinar el error. Sin embargo, en ocasiones no será así; supongamos que ha pasado varios minutos estudiando cada instrucción y no ha encontrado nada anómalo.

En estos casos usted debe recurrir a un método más eficaz para investigar el funcionamiento de la máquina. Este método se llama "rastreo del programa" (programme tracing) y consiste en imaginar que usted es el ordenador, y ejecutar cada instrucción hasta detectar alguna incompatibilidad con las reglas del lenguaje BASIC. Para aplicar este método se precisa paciencia y, sobre todo, "desconectar" la propia inteligencia, a fin de ejecutar cada instrucción como si fuera un robot rutinario, evitando desviarse hacia posibles conclusiones ni generalizaciones de ningún tipo; tomar atajos significa dejar el rastreo.

Para ejecutar el programa como lo hace el ordenador, necesita conocer cómo funciona. Supongamos que nos fuera posible "congelar" el funcionamiento del C-64 entre dos comandos a mitad de la ejecución de un programa y pudiéramos abrir su tapa y ver lo que hay dentro. Lo que veríamos sería (*):

Primero, el programa, almacenado en la memoria de manera muy parecida a como usted lo mecanografió.

Segundo, las variables del programa usadas hasta ese punto. Cada variable ocupa cierto espacio de memoria y contiene un valor, que puede ser un número o una cadena.

* Es obvio que si usted levantara realmente la tapa del ordenador lo único que vería serían unos chips de silicio y otros componentes, de cuya observación no podría deducir nada.

En tercer lugar, vería que el ordenador deja constancia del punto en que se halla la ejecución del programa en todo momento. Para ello, una variable llamada "puntero del programa", o "contador del programa", memoriza el número de línea de la siguiente instrucción a ejecutar.

Después de esta inspección del interior de la máquina, "descongelemos" ésta el tiempo suficiente para que ejecute un comando. Naturalmente, este comando seleccionará el que se había entrado en la variable del puntero del programa. Cuando mire de nuevo, después de la ejecución de una instrucción adicional, se habrá producido una serie de cambios. Por ejemplo:

(a) si se trataba de un comando PRINT, en la pantalla habrá aparecido algo;

(b) si era un comando LET, se habrá creado una nueva variable, de ser necesario, a la que se habrá adjudicado un valor, o se habrá variado el valor de una variable existente;

Tanto el comando LET como PRINT cambiarán también el número almacenado en el puntero, que pasará a ser el de la siguiente línea del programa, a ejecutar cuando se reanude el funcionamiento.

(c) una instrucción GOTO no exhibirá nada en la pantalla ni modificará ninguna variable, con excepción de la cifra contenida en le puntero, que pasará a ser la indicada en el comando especificado en GOTO. Así, la instrucción

130 GOTO 270

hará que en el puntero del programa el número existente cambie a 270.

(d) una instrucción IF operará de manera análoga, con la diferencia de que primero elaborará la condición: si es verdadera, el contador del programa se situará al número de línea especificado, como si se tratara de un comando GOTO, mientras que si es falsa el contador se situará al número de línea siguiente. Observe las dos instrucciones listadas a continuación:

120 IF X = 5 THEN 170

130 PRINT "NO"

Si X es igual a 5, la condición es verdadera y el contador del programa cambiará a 170; si es diferente, el puntero simplemente avanzará a 130.

(e) el comando STOP indica que el programa ha terminado mediante la indicación del mensaje BREAK. No hay necesidad de investigar el programa más allá de este punto.

Para realizar las operaciones del programa de manera análoga a como las ejecuta el ordenador, es esencial examinar detenidamente los siguientes parámetros: el programa, las variables, la imagen de la pantalla y el contador del programa. Un buen método de realizar este examen es utilizar un diagrama de rastreo del programa (programme tracing chart), que se va anotando en sucesivas hojas de papel. El diagrama puede construirse como en el modelo ilustrado a continuación.

PUNTERO DE PROGRAMA: 10

VARIABLES:

| *PANTALLA* | *PROGRAMA* |
|---|---|
| | 10 A = 5 |
| | 20 PRINT"ALPHA = ";A |
| | 30 A = A ★ 3 |
| | 40 B = A + 37 |
| | 50 PRINT"BETA = ";B |
| | 60 STOP |

El programa a rastrear se anota en la columna de la derecha. En la línea superior se anota el valor inicial del puntero del programa —es decir, el número de línea del primer comando a analizar—. Asegúrese de que para el rastreo utiliza una copia exacta del programa o parte del programa que contiene la dificultad; de lo contrario el rastreo será una pérdida de tiempo.

Ahora puede empezar. El contador del programa es "10", por lo que se debe interpretar la instrucción nº 10, o sea A = 5. A = 5 es una instrucción LET. En la columna "Variables" no hay ninguna A, por lo que debe anotar en la misma una "A", con el valor 5, que puede separar por dos puntos. Luego cambie el puntero al siguiente número de línea y tache el valor anterior.

PUNTERO DE PROGRAMA: ~~10~~ 20

VARIABLES A: 5

| *PANTALLA* | *PROGRAMA* |
|---|---|
| | 10 A = 5 |
| | 20 PRINT"ALPHA = ";A |
| | 30 A = A ★ 3 |
| | 40 B = A + 37 |
| | 50 PRINT"BETA = ";B |
| | 60 STOP |

Interprete el siguiente comando de forma análoga. Conviene que olvide la finalidad del programa y todo conocimiento que tiene del

orden de ejecución de las instrucciones; ejecute el comando 2Ø sólo por estar indicado en el puntero. Este comando es PRINT y determinará, evidentemente, la puesta en pantalla del mensaje "ALPHA = 5".

Anote, pues, dicho mensaje en la columna PANTALLA y avance el contador, es decir:

CONTADOR DE PROGRAMA ~~1Ø~~ ~~2Ø~~ 3Ø

VARIABLES A: 5

| *PANTALLA* | *PROGRAMA* |
|---|---|
| ALPHA = 5 | 1Ø A = 5 |
| | 2Ø PRINT"ALPHA = ";A |
| | 3Ø A = A★3 |
| | 4Ø B = A + 37 |
| | 5Ø PRINT"BETA = ";B |
| | 6Ø STOP |



La siguiente instrucción asigna un nuevo valor a la variable A. Para calcular la expresión A★3, multiplica el anterior valor de A(5) por 3 y anota el resultado, de esta forma:

A: 5,15

El siguiente comando crea una nueva variable. Continúe el rastreo hasta que llegue a STOP. El resultado final será:

CONTADOR DE PROGRAMA: ~~1Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ ~~5Ø~~ 6Ø

VARIABLES A: 5,15 B: 52

| *PANTALLA* | *PROGRAMA* |
|---|---|
| ALPHA = 5 | 1Ø A = 5 |
| BETA = 52 | 2Ø PRINT"ALPHA = ";A |
| BREAK IN 6Ø | 3Ø A = A★3 |
| READY. | 4Ø B = A + 37 |
| | 5Ø PRINT"BETA = ";B |
| | 6Ø STOP |

A continuación se lista otro programa con un bucle, cuyo rastreo se describe seguidamente a título de ejemplo:

```
1Ø P = 1
2Ø PRINT P;P★P★P
3Ø P = P + 1
4Ø IF P<4 THEN 2Ø
5Ø STOP
```

El rastreo de este programa hasta la línea 30 es relativamente fácil:

CONTADOR DE PROGRAMA ~~1Ø~~ ~~2Ø~~ ~~3Ø~~ 4Ø

VARIABLES P: 1,2

| *PANTALLA* | *PROGRAMA* |
|---|---|
| 1 1 | 1Ø P = 1 |
| | 2Ø PRINT P;P★P★P |
| | 3Ø P = P + 1 |
| | 4Ø IF P<4 THEN 2Ø |
| | 5Ø STOP |

La siguiente instrucción es la nº 4Ø, un comando IF. Para imitar al ordenador, evalúe la condición P<4. Puesto que el valor actual de P es 2 (esto es lo que vemos en la columna Variables) y 2 es, evidentemente, menor que 4, la condición es verdadera. Por consiguiente, usted pone 2Ø como nuevo valor del contador. Su diagrama tendrá ahora el siguiente aspecto:

CONTADOR DE PROGRAMA ~~1Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ 2Ø

VARIABLES P: 1,2

| *PANTALLA* | *PROGRAMA* |
|---|---|
| 1 1 | 1Ø P = 1 |
| | 2Ø PRINT P;P★P★P |
| | 3Ø P = P + 1 |
| | 4Ø IF P<4 THEN 2Ø |
| | 5Ø STOP |

Usted continúa el rastreo hasta que finalmente la condición es falsa, en cuya ocasión el programa pasa a la siguiente instrucción y, por lo tanto, se para, al encontrar STOP. El resultado final del diagrama de rastreo es:

CONTADOR DE PROGRAMA: ~~1Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ 5Ø

VARIABLES P: 1,2,3,4

| *PANTALLA* | *PROGRAMA* |
|---|---|
| 1 1 | 1Ø P = 1 |
| 2 8 | 2Ø PRINT P;P★P★P |
| 3 27 | 3Ø P = P + 1 |
| BREAK IN 5Ø | 4Ø IF P<4 THEN 2Ø |
| READY. | 5Ø STOP. |

# EXPERIMENTO 8·1

Practique ahora la técnica de rastreo de programas utilizando las siguientes secuencias. Efectúe sus anotaciones con un lápiz, a fin de poder borrarlas cuando se equivoque.



(a)

CONTADOR DE PROGRAMA

VARIABLES

*PANTALLA*

*PANTALLA*

```
1Ø X = 5
2Ø Y = 7
3Ø Z = X + Y
4Ø W = Y – X
5Ø PRINT X;Y;Z;W
6Ø STOP
```

(b)

CONTADOR DE PROGRAMA

VARIABLES

*PROGRAMA*

*PROGRAMA*

```
1Ø Q = 1
20 PRINT "ME
QUIERE"
30 PRINT "NO ME
QUIERE"
4Ø Q = Q + 1
5Ø IF Q<3 THEN 3Ø
6Ø STOP
```

Cuando haya terminado estos dos ejercicios de rastreo compruebe los resultados con los indicados en el Apéndice B.

*Fin del Experimento 8.1*

# EXPERIMENTO 8·2



¿Cómo puede utilizarse el rastreo, para encontrar errores? La explicación reside en adoptar una postura intermedia entre robot y persona, capaz de razonar. Se pueden ir ejecutando las instrucciones mecánicamente como lo haría la máquina y después comparar los resultados obtenidos con lo que usted esperaba obtener, y así poder detectar los puntos donde existan errores. A continuación se da un ejemplo de cómo detectar la diferencia entre los resultados obtenidos con el rastreo y los deseados al escribir el programa.

El primer ejemplo es un programa que dé una tabla de multiplicar por 12. Usted quiere que en la pantalla aparezca la siguiente tabla:

TABLA DE MULTIPLICAR DEL DOCE

1 ★ 12 = 12

2 ★ 12 = 24

3 ★ 12 = 36

(y así sucesivamente hasta)

12 ★ 12 = 144

Usted incluye en el programa todos los elementos necesarios: un bucle, una instrucción que exhiba el título de la tabla y otra que muestre las sucesivas líneas de la tabla, es decir:

```
10 PRINT"TABLA DE MULTIPLICAR DEL DOCE"
20 P = 1
30 P = P + 1
40 IF P<13 THEN 30
50 PRINT P; "★12 = ";P★12
60 STOP
```

Sin embargo, cuando usted ejecute el programa, el resultado en la pantalla es:

```
TABLA DE MULTIPLICAR DEL DOCE
13 ★ 12 = 156
BREAK IN 60
READY
```

No es en modo alguno lo que usted esperaba. Puede que usted vea enseguida el error, pero supongamos que no sea así y decide realizar un rastreo. Al cabo de poco ha llegado a los siguientes resultados:

CONTADOR DE PROGRAMA ~~10 20 30 40 30 40~~ 30

VARIABLES P: 1,2,3,4

| *PANTALLA* | *PROGRAMA* |
|---|---|
| TABLA DE MULTIPLICAR POR DOCE | 10 PRINT"TABLA DE MULTIPLICAR POR DOCE" |
| | 20 P = 1 |
| | 30 P = P + 1 |
| | 40 IF P<13 THEN 30 |
| | 50 PRINT P;"★12 = ";P★12 |
| | 60 STOP |

Al llegar a ese punto usted se da cuenta de que el valor de la variable P va incrementándose hasta 12 sin que nada aparezca en la pantalla. Es, pues, evidente que el comando PRINT debería estar dentro del bucle, en vez de fuera. Su ubicación correcta es entre las instrucciones 20 y 30. A su vez el comando IF debe cambiarse, para que la ejecución salte a PRINT. Una rápida edición del programa nos da la siguiente secuencia:

```
10PRINT"TABLA DE MULTIPLICAR DEL DOCE"
20 P=1
30 P=P+1
40 PRINT P;"*12=";P*12
50 IF P<13 THEN 30
60 STOP
```

El rastreo del programa es una técnica extremadamente útil, a condición de que se tenga la paciencia de ejecutar cada instrucción en todos sus efectos. Si en vez de esta ejecución sistemática y mecánica trata de predecir el resultado de cada instrucción por simple intuición, lo más probable es que se equivoque de nuevo como hizo al escribir originalmente el programa. Un rastreo parcial a menudo no da ninguna clave de la anomalía de un programa.

No obstante, en ciertas circunstancias el rastreo resulta inaplicable o infructuoso. Conviene que usted las conozca por lo cual se relacionan a continuación:

- Si el programa es muy largo, un rastreo sistemático exigirá demasiado tiempo. En tales casos deben aplicarse otras técnicas, que se describen más adelante.
- Si usted cree que no puede equivocarse, el rastreo y los programas que haya escrito le servirán de poco. Al escribir la última línea de un programa a veces se tiene la sensación de que esta vez no va a haber ningún error; la razón de ello es que no se es consciente de haberse equivocado en ningún punto, lo que no descarta la equivocación; es preferible adoptar la actitud opuesta: "sin duda hay algún error; vamos a ver dónde está".
- Si hay algún punto fundamental del BASIC que usted no haya comprendido, el realizar un rastreo le servirá de poco. Imaginemos un error de interpretación muy grande, como suponer que el signo "—" signifique sumar. Si Vd. escribe el siguiente programa para "sumar" dos números:

```
10 A=34
20 B=19
30 PRINT "A =";A
40 PRINT "B =";B
50 PRINT "A MAS B =";A—B
60 STOP
```

Cree que esto significa sumar

Cuando usted ejecuta el anterior programa, se encuentra con que la pantalla ha adquirido el siguiente aspecto:

```
A = 34
B = 19
A MAS B = 15
BREAK IN 60
READY
```

El resultado es, evidentemente erróneo; sin embargo, en tanto uno crea que "—" significa "sumar" no se detectará el error.

Naturalmente, los errores de interpretación de los términos o reglas de uso del lenguaje BASIC son más sutiles que ese ejemplo, y si los resultados de un rastreo difieren de los exhibidos en la pantalla, es evidente que hay algún defecto en la técnica de rastreo aplicada. Si usted no encuentra solución al problema, consulte a algún experto en BASIC que usted conozca; si ello no es posible, vuelva a leer este curso desde un principio, para ver qué punto interpretó incorrectamente; una revisión sistemática de lo tratado hasta aquí le dará casi invariablemente la clave del problema.

Claro está, también, es posible que el ordenador tenga una avería; sin embargo, ello es extremadamente raro, ya que las máquinas C-64 son de gran resistencia y fiabilidad, y de averiarse la anomalía suele ser obvia: por ejemplo, el cursor no aparece en la pantalla cuando se conecta el ordenador o no es posible cargar programas desde el cassette. En la práctica no debe pensarse que la incongruencia de un programa se debe a la máquina; sin embargo, si usted ha agotado las posibilidades de detección del error, lleve el ordenador al distribuidor C-64, para que compruebe dónde está la anomalía; explique exactamente en qué consiste el defecto, incluyendo una copia del programa que no puede ejecutarse.

He aquí dos programas que incluyen errores para que usted los examine con la técnica de rastreo para su detección y corrección:

(a) Este programa tiene como fin convertir la medida anglosajona de capacidad "gallon" a litros, en la gama de 1 a 10 gallons (1 gallon = 4'5 litros).

```
10 PRINT"GALLONS","LITROS"
20 G = 1
30 PRINT G, 4.5*G
40 G = G + 1
50 IF G>11 THEN 30
60 STOP
```

(b) Este programa se ha concebido para exhibir un triángulo de estrellas, como el descrito en el problema 1 de la Unidad 7, pero no consigue su propósito. ¿Por qué?

```
10 A$ = "*"
20 PRINT A$
30 A$ = "**"
40 IF A$<>"**********"THEN 20
50 STOP
```

*Fin del Experimento 8.2.*



# EXPERIMENTO 8·3

El programa UNI8PROG tiene como fin exhibir la tabla de multiplicar por 7, pero contiene varios errores. Cárguelo en la memoria del C-64, detecte y corrija los errores. Compruebe si sus soluciones son las que se indican en el Apéndice B.

*Fin del Experimento 8.3*

# UNIDAD: 9


| | | |
|---|---|---|
| EXPERIMENTO 9.1 | PAG. | 65 |
| EXPERIMENTO 9.2 | | 68 |
| EXPERIMENTO 9.3 | | 69 |

# EXPERIMENTO 9·1

Dibujemos ahora algunas figuras en la pantalla, pero esta vez dejemos que el C-64 haga todo el trabajo.

Usted recordará que en las unidades de estudio 2 y 3 decíamos que era posible dibujar figuras en la pantalla utilizando diversas "funciones" de control:

- El desplazamiento del cursor en cuatro direcciones.
- La selección de dieciseis diferentes colores.
- La inversión del color de los caracteres y del fondo (conexión y desconexión del dispositivo de inversión).
- Restitución del cursor a su posición original en la esquina superior izquierda.
- Borrado de la pantalla.

Estas funciones exigen la pulsación de más de una tecla, principalmente de SHIFT o CTRL

No habrá olvidado usted que puede seleccionar el color del bastidor y del fondo de la pantalla con dos instrucciones POKE, seguidas de un número de código.

El C-64 puede dibujar en la pantalla cualquier figura por sí solo, simplemente bajo control de programa. En cada programa pueden incorporarse todas las funciones de control de la pantalla: es posible seleccionar cualquier color para los caracteres, borrar la pantalla en cualquier momento y desplazar el cursor interno (invisible al usuario) a cualquier posición usando las funciones de control del cursor.

Naturalmente, el C-64 realiza todo eso por sí mismo, sólo si las pertinentes instrucciones se incorporan en un programa. Incluir funciones de control de la pantalla en un comando es fácil: simplemente se escriben en las cadenas, junto al resto de caracteres a exhibir. Al principio todo ello puede que parezca algo desconcertante, porque, si en una cadena se incluye una función de borrado de la pantalla, va a borrarse todo lo que está visualizado ¿o no es así? En realidad no ocurre este borrado general, como se verá en el siguiente Experimento.

¿Recuerda usted que en la Unidad 2 dijimos que no usara las comillas, porque podían dar resultados inesperados? A continuación veremos el efecto del signo comillas y sus aplicaciones.

Cuando se empieza a escribir una instrucción (por ejemplo después de pulsar RETURN o cuando en la pantalla aparece el mensaje READY). El C-64 está en Modo Normal. Las funciones de control tales como selección de colores o desplazamiento del cursor operan en dicho modo como sería de esperar. Sin embargo, cuando se pulsa comillas para marcar el comienzo de una cadena, la máquina cambia al Modo Comillas; en este Modo, los caracteres corrientes, tales como letras o signos gráficos, son tratados de la forma normal, pero las funciones de control no son obedecidas, sino que se incorporan a la cadena como caracteres especiales, principalmente como letras y signos gráficos sobre fondo de color invertido. La máquina vuelve a cambiar a Modo Normal cuando se cierran las comillas (terminando la cadena) o se pulsa la tecla RETURN

Conecte el C-64, escriba el signo comillas y pulse todas las teclas de funciones, sucesivamente.

Vea el aspecto que adquiere en la pantalla cada función de control y anótelo en la siguiente tabla, en la columna de la derecha.

| *Función* | *Teclas a pulsar* | *Símbolo en la pantalla* |
|---|---|---|
| Borrado pantalla | SHIFT y CLR HOME | |
| Cursor en posición original | CLR HOME | |
| Cursor hacia arriba | SHIFT y CRSR ↕ | |
| Cursor hacia abajo | CRSR ↕ | |
| Cursor a la izquierda | SHIFT y CRSR ↔ | |
| Cursor a la derecha | CRSR ↔ | |
| Negro | CTRL y BLK | |
| Blanco | CTRL y WHT | |
| Rojo | CTRL y RED | |
| Cián | CTRL y CYN | |
| Morado | CTRL y PUR | |
| Verde | CTRL y GRN | |
| Azul | CTRL y BLU | |
| Amarillo | CTRL y YEL | |
| Conexión inversión | CTRL y RVS ON | |
| Desconexión inversión | CTRL y RVS OFF | |

Veamos ahora algunos de estos controles en acción. Primero asegúrese de que su receptor de televisión está debidamente ajustado para color, utilizando el programa CARTA DE AJUSTE de ser necesario.

Luego escriba la palabra "EDINBURGO" de modo que aparezca en amarillo:

PRINT" CTRL y YEL EDINBURGO"

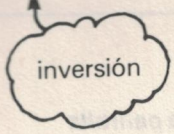
(Apretar CTRL mientras se pulsa YEL)



Dream commodore

Lo que aparece realmente en la pantalla (todo todavía en azul) es:

PRINT" π EDINBURGO". (el símbolo en color




invertido π es el código para el color amarillo).

Ahora, pulse la tecla RETURN . La palabra EDINBURGO aparece ahora en la pantalla en amarillo.

Este experimento ilustra el principio muy claramente: cuando una función de control se escribe en el teclado dentro de una cadena, no ejerce su acción cuando se mecanografía, sino que es ejecutada cuando la cadena es mostrada por el ordenador.

Usted se habrá fijado en que el cursor centelleante ha quedado en color amarillo. Puede cambiarlo a negro o azul, el color que prefiera, pulsando la correcta función de control, sin comillas.

Cualquier comando PRINT que da un cambio de color puede incorporarse en un programa, al igual que cualquier otra instrucción. Escriba y ejecute la siguiente secuencia:

10 PRINT " CTRL y GRN GLASGOW"
20 PRINT " CTRL y RED INVERNESS"
30 PRINT " CTRL y PUR ST. CTRL
y YEL ANDREWS"
40 STOP

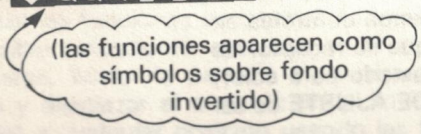
En la instrucción 30 vemos que es posible poner más de una función de control en una misma cadena.

Las funciones de control de la pantalla y del cursor pueden también incluirse en cadenas. Escriba:

PRINT " SHIFT y CLR HOME CRSR CRSR CRSR CRSR
CRSR CRSR CTRL y RED PARIS"

En la pantalla aparece:

PRINT " ♥ QQQ ] ] ] £ PARIS"




Las funciones de control son ejecutadas cuando usted pulsa la tecla RETURN : la pantalla se borra, el cursor baja tres líneas y se desplaza lateralmente tres columnas. Entonces la palabra PARIS APARECE en rojo, a mitad de camino hacia el centro de la pantalla. Véalo usted mismo.

En general usted puede programar la aparisiempre que sea necesario para restituir C-64 a cualquier punto de la pantalla, incluyendo en la cadena el correcto número de desplazamientos del cursor.

Cuando usted quiere que el ordenador dibuje una figura en la pantalla, no quiere que la imagen se eche a perder con la presencia de la palabra READY y el cursor centelleante. Una forma de evitar esto es mediante un bucle de parada, es decir un comando GOTO que produce un salto hacia el mismo: cuando el ordenador llega a esta instrucción, no mostrará la palabra READY hasta que se pulse RUN STOP . El siguiente programa, por ejemplo, exhibirá la palabra LONDON en blanco en el centro de una pantalla negra:

10 POKE 53281,0: POKE 53280,1
(10 veces)
20 PRINT " SHIFT y CLR HOME CRSR . . . CRSR
CRSR . . . . CRSR CTRL y WHT LONDON"
(8 veces)
30 GOTO 30

Escriba este programa, ejecútelo y luego páre-lo con la tecla RUN STOP . La pantalla seguirá siendo negra y el cursor blanco, pero puede volver fácilmente a los colores normales apretando la tecla RUN STOP y, sin soltarla, pulsando RESTORE . En realidad usted puede apretar estas dos teclas siempre que sea necesario para restituir el C-64 a funcionamiento normal; es preferible a desconectar el ordenador y volverlo a conectar, porque no implica la pérdida del programa cargado en la memoria.

Como breve ejercicio, haga que el C-64 exhiba palabras y pautas de signos gráficos de diferentes colores y en diferentes posiciones de la pantalla. Recuerde que la función SHIFT y CLR HOME borra la pantalla, por lo que si su programa contiene una serie de instrucciones PRINT, sólo la primera debería empezar con esta función, aún cuando otras pueden empezar apretando CLR HOME por sí sola.

*Fin del Experimento 9.1*

# EXPERIMENTO 9·2

Seguramente usted sabe que los relojes modernos son controlados por cristales de cuarzo, que les confieren una extremada precisión por largos períodos de tiempo. El C-64 contiene también un cristal de cuarzo que vibra varios millones de veces por segundo y que es utilizado, entre otros fines, para controlar un reloj digital interno. Este reloj carece de esfera y es tratado como una variable de cadena, de modo que usted puede exhibir en la pantalla la hora siempre que lo desea. El nombre de esa variable horaria es TI$.

Cuando usted pone en marcha un reloj, tiene que ponerlo en hora. El reloj incorporado en el C-64 no es ninguna excepción a esta regla. Para ajustarlo a la hora, pulse un comando como:

TI$ = "193746" RETURN

Este códico ajustará la hora a 7.37 y 46 segundos de la tarde.

Si quiere ajustar el reloj interno del C-64 con precisión, es preferible esperar que sea la hora exacta, como por ejemplo las 9 de la mañana en la radio. Antes de que dé la hora, escriba en el teclado:

TI$ = "Ø9ØØØØ"

y cuando oiga la última señal horaria, indicando que son las nueve en punto, pulse RETURN

Cuando el reloj del C-64 se ha puesto a la hora, mantiene el tiempo correcto —con una aproximación de algunos segundo al día— hasta que se desconecta el ordenador.

Para ver la hora en la pantalla, escriba simplemente TI$ en un comando PRINT sin comillas.

Ahora, ajuste la hora del C-64 a su reloj (no importa si no es absolutamente exacto). Luego pida al C-64 que indique la hora en la pantalla, con el pertinente comando PRINT, y vea como van sucediéndose los segundos.

Ahora muestre la hora de modo fijo, con el siguiente bucle:

```
1Ø PRINT TI$
2Ø GOTO 1Ø
```

Pare el programa, espere unos momentos y vuelva a ejecutarlo: verá que sigue marcando la hora correcta, es señal de que el reloj no se ha parado.

Este método de indicar la hora no es muy sofisticado. Usted puede convertir al C-64 en un respetable reloj digital con un programa como el siguiente:

comando 10: Selecciona un fondo de pantalla amarillo, enmarcado por un borde de morado

comando 20: Borra la pantalla

comando 30: Restituye el cursor a su posición original, luego lo baja 9 hileras y lo avanza 6 columnas; todo ello en una misma línea.

comando 40: Muestra el valor de TI$

comando 50: Salta otra vez al comando 30

Escriba el anterior programa en el espacio delimitado por el rectángulo; luego mecanografíelo en el teclado del C-64 y ensaye su ejecución. Si no consigue crear un programa eficaz, vea la versión correcta en el apéndice B, pero no continúe hasta haberlo estudiado detenidamente y comprendido cómo opera.

*Fin del Experimento 9.2*



Dream Commodore

# EXPERIMENTO 9·3



Los bucles de control a menudo pueden utilizarse para dibujar formas en la pantalla. Supongamos que usted quiere un bloque de 10 × 10 puntos rojos en la esquina superior izquierda de la pantalla. Puede conseguirlo mostrando diez líneas, cada una con diez signos ●:

```
1Ø PRINT" SHIFT y CLR HOME ";
2Ø J = 1
3Ø PRINT" CTRL y RED ● ● ● ● ● ●
3Ø ● ● ● ●"
4Ø J = J + 1
5Ø IF J<11 THEN 3Ø
6Ø GOTO 6Ø
```

En este programa se combinan varias de las ideas que hemos tratado ya en anteriores unidades de estudio. El punto y coma al final de la línea 1Ø impide que la máquina empiece una nueva línea después de borrar la pantalla, con lo que la primera línea de puntos rojos aparece en la hilera superior. Las instrucciones 2Ø a 5Ø forman un bucle de control, y la nº 6Ø un bucle de paro.

Mecanografíe este programa y ejecútelo hasta que quede parado ejecutando permanentemente la instrucción nº 6Ø. Luego pare el programa y vea lo que pasa si:

(a) cancela el punto y coma después de " CLR HOME ";

(b) cambia el número 11 de comando 5Ø por cualquier otro valor (por ejemplo 15).

(c) cancela el comando 6Ø.

Para efectuar estas correcciones, liste y edite el programa como de costumbre. Antes de empezar, no se olvide de restituir el cursor al color negro o azul.

Si, en cambio, deseamos obtener un bloque sólido de color, usamos espacios sobre fondo invertido, Vea lo que ocurrirá si cambia la instrucción 3Ø del anterior programa por la siguiente:

```
PRINT " CTRL y RED CTRL y
RVS ON ← 1Ø espacios →"
```

Luego vuelva a ejecutar el programa.

¿Cómo podemos conseguir que en la pantalla aparezca más de un bloque de color en una misma figura? Para ello, situar el cursor en la primera línea del área y rellenarla, sin interferir con el color que ya esté en la pantalla. Consideremos dos ejemplos:

(a) Crear un bloque azul 1Ø × 1Ø exactamente debajo del bloque rojo:

La mitad inferior de la pantalla está vacía, por lo que no hay riesgo de estropear ninguna imagen ya existente en dicha parte. Además, después de producir el anterior bloque rojo, el cursor estará en el lugar a requerido. Podemos ampliar el anterior programa añadiéndole las siguientes línea:

```
6Ø J = 1
7Ø PRINT" CTRL y BLU CTRL y
RVS ON ← 1Ø espacios →"
8Ø J = J + 1
9Ø IF J<11 THEN 7Ø
1ØØ GOTO 1ØØ
```

Observe que hemos desplazado el bucle de parada al final del programa, como era antes el caso. J se usa como una variable de control tanto en el bucle rojo como en el azul: Esto es posible, puesto que el bloque rojo queda terminado antes de empezar el azul, por lo que no se pide a J que desempeñe dos tareas al mismo tiempo.

(b) Crear un bloque negro 1Ø × 1Ø junto al rojo:

Hemos de empezar esta vez en la línea superior de la pantalla, por lo que al crear el área negra hemos de prestar cuidado para no estropear el bloque rojo existente en la esquina superior izquierda. Para ello podemos situar primero el cursor en su posición original y trazando 1Ø líneas, cada una empezando con 1Ø movimientos del cursor hacia la derecha para que salte el bloque rojo. La ampliación del programa será, pues:

```
1ØØ PRINT" CLR HOME ";
11Ø J = 1
12Ø PRINT" CRSR . . . . . CRSR
CTRL y BLK CTRL y
RVS ON ← 1Ø espacios →"
13Ø J = J + 1
14Ø IF J<11 THEN 12Ø
15Ø GOTO 15Ø
```

1Ø veces

Escriba ahora consecutivamente en una hoja de papel las tres secuencias del programa, mecanografíe éste en el teclado y ejecútelo con RUN. Observe que contiene tres bucles separados, que se ejecutan. Uno después del otro.

Extienda ahora usted el programa para añadir un bloque morado debajo del negro...

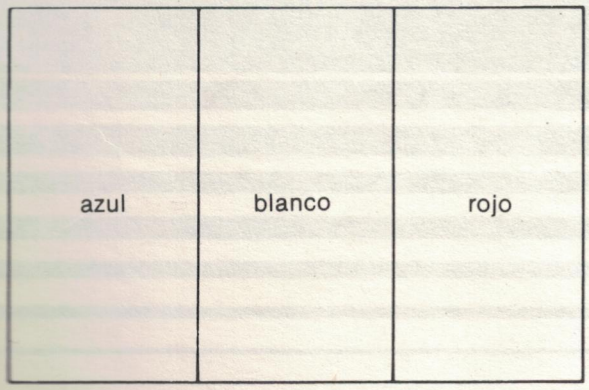
Como ejercicio final, escriba programas para exhibir banderas nacionales simples. Usted tendrá que concentrarse en su trabajo, porque encontrará algunos riesgos. Recuerde bien los siguientes puntos:

- El significado normal del separador punto y coma (;) al final de un comando PRINT es "no empezar nueva línea". Usted ya sabe que si pide al C-64 que coloque un carácter o signo gráfico en la posición extrema derecha de una línea, el cursor pasará automáticamente a la siguiente línea; por consiguiente, si quiere que un bloque de color no deje líneas intermedias en blanco, debe terminar cada línea con punto y coma.

- No se puede usar un comando PRINT, para escribir un caracter en la parte inferior derecha de la pantalla, sin que se desplace hacia arriba todo lo que hay en la misma. Por consiguiente, como se ha explicado, la única forma de asignar el color deseado a dicha posición consiste en seleccionar el color de fondo de la pantalla, para que coincida con el que nosotros queremos en esa posición.

Prepare las figuras a programar cuidadosamente, dibujándolas en papel cuadriculado. Cuando empiece a escribir los programas en cuestión, no se desanime si se equivoca varias veces, e insista hasta conseguir el resultado deseado. Recuerde que se aprende creando programas correctos, no programas que no funcionan; es decir, en este caso uno no aprende de los errores, sino de los éxitos.

Para practicar, le explicaremos un programa para crear la bandera francesa:




Asignemos a la franja central blanca 14 espacios de anchura y 13 a las otras dos, o sea 13 + 14 + 13 = 40. Los colores iniciales pueden ser rojo para el fondo y negro para el borde.

Podemos, así, constituir la bandera exhibiendo 25 líneas, cada una con trece cuadrados blancos y catorce azules. Recuerde que la última línea ha de ser diferente, para que no vaya seguida de nueva línea. Podemos incluir las primeras 24 líneas dentro de un bucle de control, pero la última línea ha de poseer su propio comando.

Escribimos, así, el siguiente programa:

```
10 POKE 53280,0: POKE 53281,2
20 PRINT" [SHIFT] y [CLR HOME] ";
30 J=1
40 PRINT" [CTRL] y [RVS ON] [CTRL] y
   [BLU] ←13 espacios→ [CTRL] y
   [WHT] ←14 espacios→"
50 J=J+1
60 IF J<25 THEN 40
70 PRINT" [CTRL] y [RVS ON] [CTRL] y
   [BLU] ←13 espacios→ [CTRL] y
   [WHT] ←14 espacios→";
80 GOTO 80
```

Ejecute este programa y estudielo cuidadosamente hasta que comprenda a la perfección cada símbolo. Ensaye la programación de algunas banderas, evitando las que contienen elementos diagonales. Pruebe, por ejemplo, con la bandera islandesa, ilustrada en la pág. 19. Compruebe su programa con el listado en el Apéndice B.

| *Fin del Experimento 9.3* | |
|---|---|

El programa para comprobar su dominio de las materias tratadas en esta unidad de estudio lleva por título UNIT9QUIZ.

# UNIDAD: 10


EXPERIMENTO 10.1 PAG. 74

EXPERIMENTO 10.2 75

En las anteriores unidades de estudio hemos visto como las instrucciones de un programa pueden ejecutarse varias veces de forma repetida, con sólo escribirlas una vez. Ello se consigue mediante los bucles.

A mayor escala un proceso análogo ocurre con todos los programas de uso práctico, ya que se usan repetidamente, incorporando o extrayendo datos y todo tipo de información. A este fin, los programas se transfieren a cintas o paquetes ROM, para su ejecución cuando se necesitan. Un ejemplo de ello son los programas en cinta cassette.

Consideramos primero los programas que usted ha ido escribiendo a lo largo de este curso. Un inconveniente común a todos ellos es que, no importa las veces que se ejecuten, producen siempre el mismo resultado. Su utilidad es, pues, muy limitada.



Un ejemplo de este inmovilismo es el programa con que calculábamos la conversión entre libras esterlinas y liras italianas. El programa era:

```
10 PRINT"£","LIRAS"

20 PRINT

30 PS = 5

40 PRINT PS, 2350 * PS

50 PS = PS + 5

60 IF PS<80 THEN 40

70 STOP
```

El día que se escribió el programa, el cambio era de 2.350 liras, por lo que los resultados eran correctos, pero supongamos que debido a las fluctuaciones monetarias el cambio haya descendido a 2.175; con ello el programa queda automáticamente anticuado. Naturalmente, se podría actualizar modificando la línea 40, de modo que tuviera el siguiente valor:

```
40 PRINT PS, 2175 * PS
```

Desafortunadamente, esta solución tendría poco éxito en la práctica; en primer lugar, los usuarios de ordenadores en los bancos no son programadores, y, en segundo lugar, aunque lo fueran, prestarían escaso interés a un programa de este tipo.

Para hacer que los programas sean más flexibles y adaptables a las necesidades cambiantes de la vida diaria, necesitamos una función adicional, que permita al usuario entrar información en respuesta a la cual el programa dé soluciones diferentes; en nuestro caso, supongamos que el programa de conversión de moneda permitiera al usuario entrar el tipo de cambio cada vez que fuera necesario o cada vez que se usara: en este último caso, podría ser operado por el personal de cambio en bancos de todo el mundo, sin ser afectado por las oscilaciones del mercado.

Suponga que está escribiendo un programa que ha de ser usado por otra persona; lo primero que debe decidir es qué cantidades han de quedar indefinidas, para que las entre el usuario. En el anterior ejemplo, el tipo de cambio era evidentemente una cantidad de esta categoría: desconocida para el programador, pero conocida por el usuario. Por consiguiente, en vez de una cantidad, el programa incluye una variable; por ejemplo, llamaremos a la variable cambio de moneda, RE (Rate of exchange). De este modo usted puede escribir sus programas usando nombres simbólicos en vez de los valores reales (que se desconocen de antemano). En tal caso la línea 40 de nuestro modelo sería:

```
40 PRINT PS, RE * PS
```

Naturalmente, no lo hemos dicho todo: para que la máquina sepa las cantidades que se desea asignar a las variables se requiere una instrucción especial, denominada INPUT. Esta palabra clave va seguida entonces del nombre o nombres de las variables. Al ejecutarse la instrucción INPUT, se interrumpe la marcha del programa en espera de que el usuario mecanografíe un valor, que se almacena entonces en la variable nombrada. Con ello puede ejecutarse el resto del programa en que se usa esta variable. Sin embargo, para que el usuario sepa que la máquina espera los datos en cuestión es preciso que en la pantalla aparezca el pertinente mensaje, lo que se consigue con el comando PRINT.

# EXPERIMENTO 10·1

Estudie detenidamente el siguiente programa:

```
3 PRINT"ENTRE EL CAMBIO"
4 PRINT"DE HOY ENTRE"
5 PRINT"LA £ Y LIRA"
6 INPUT RE
1Ø PRINT "£","LIRA"
2Ø PRINT
3Ø PS = 5
4Ø PRINT PS,RE★PS
5Ø PS = PS + 5
6Ø IF PS<8Ø THEN 4Ø
7Ø STOP
```

Observe como el programa no presupone un tipo de cambio específico, sino que lo representa por la variable RE en todas las instrucciones en que se cita. El programa empieza preguntando al usuario los datos que necesita.

Entre ese programa, asegúrese de que lo ha escrito correctamente y ejecútelo. Imagínese que es el usuario, un empleado de banca en el departamento de cambio, sin conocimiento de programación. En la pantalla, la máquina le pide que entre una cifra, por lo que mecanografía el cambio del día y pulsa la tecla RETURN. La respuesta de la máquina es instantánea: en la pantalla aparece una tabla de conversión con todos los datos que usted necesita para cambiar libras en liras.

Ejecute el programa varias veces, entrando cada vez un tipo de cambio diferente, para ver la velocidad con que realiza los cálculos y los exhibe.

Recupere ahora su personalidad de programador. Durante la ejecución del programa, cuando éste se paró en espera de los datos para la variable RE, dicha petición de datos adicionales fue en respuesta al comando INPUT.

El comando INPUT puede adoptar formas ligeramente diferentes. Veamos, algunos ejemplos y haremos mención de algunas reglas generales:

1. Borre lo que hay en memoria escribiendo NEW y copie el siguiente programa:

```
1Ø PRINT"COMO TE LLAMAS"
2Ø INPUT N$
3Ø PRINT"HOLA [SPACE]"; N$;"!"
```

Ejecute este programa y vea qué pasa. Es un ejemplo del uso del comando INPUT solicitando la entrada de una cadena en vez de un número. Usted puede usar una secuencia de este tipo en los comienzos de cualquier programa si desea que el ordenador haga gala de amabilidad con respecto al usuario. O si se trata de un test, puede usar el valor N$ en instrucciones como las siguientes:

```
4Ø PRINT"NO [SPACE]";N$;".VD PUEDE
   HACERLO MEJOR"
```

(Si no está seguro de lo que va a aparecer en la pantalla, mecanografíe esta instrucción de modo que se incorpore a las anteriores tres líneas y vuelva a ejecutar el programa).

2. Ensaye ahora:

```
1Ø INPUT"NOMBRE";N$
2Ø PRINT"ADIOS [SPACE]";N$
```

En este ejemplo usted ve cómo puede incluirse una breve información en el propio comando INPUT. Esa información aparece en la pantalla como guía para el usuario, inmediatamente antes del signo?. En esta aplicación, el comando 1Ø equivale a la secuencia:

```
PRINT"NOMBRE";
INPUT N$
```

Recuerde que la cadena descriptiva debe ir seguida de punto y coma.

3. Finalmente, ensaye:

```
1Ø PRINT"ESCRIBA DOS CANTIDADES
   A SUMAR"
2Ø INPUT A,B
3Ø PRINT"SUM = ";A + B
4Ø STOP
```

Ahora la instrucción INPUT está en espera de dos cantidades, que el usuario debe mecanografiar separadas por una coma o por la pulsación de la tecla RETURN es decir:

43, 19

o { 4 3
  { 19

En principio, el comando INPUT puede pedir al usuario cualquier número de variables, pero es preferible limitarse a dos, para evitar confusiones. Al escribir el comando, los nombres de las variables se separan con comas.

Después de ejecutar varias veces este programa, pretenda que no sabe que ha de entrar valores numéricos y escriba cualquier cosa, como por ejemplo.



PATO,DONALD

Cuando pide una cadena el ordenador está dispuesto a aceptar cualquier respuesta, aunque no tenga sentido; pero si solicita un número y lo que usted escribe no lo es, objetará la incongruencia con el siguiente mensaje:

REDO FROM START

o sea, "vuelva a empezar", dándole una oportunidad para que corrija la entrada.

En ocasiones usted querrá parar la ejecución de un programa en la fase INPUT, cuando el ordenador está en espera de una entrada y en la pantalla aparece el cursor. En tales casos no puede pulsar la tecla RUN STOP, que no ejercería ningún efecto, sino que debe recurrir a la tecla RESTORE. ¡Al mismo tiempo que presiona la de RUN STOP. Compruébelo.

*Fin del Experimento 10.1*

# EXPERIMENTO 10·2

Para escribir programas prácticos hay que recordar que el programador y el usuario son dos personas diferentes. Así, el usuario no listará el programa con qué trabaja para averiguar como lo hace. Para comunicar con el usuario, el programa se vale de instrucciones con las que puede mostrar mensajes en la pantalla, pero esta comunicación es en un solo sentido; por dicha razón, es esencial que se informe debidamente al usuario, explicando en la pantalla lo que se le pide y cualquier situación anómala que se pueda encontrar.

Cuando escriba un programa, imagínese que usted está observando al usuario, mientras éste ejecuta el programa. Procure prever todas las posibles anomalías e introduzca los pertinentes avisos y medidas correctivas.

Cuando termine de escribir un programa, ejecútelo como si fuera el usuario. Finalmente, pida a algún amigo o familiar que haga de "conejillo de indias", para ver como reacciona en diversas situaciones de la ejecución del programa: si no entiende los mensajes o no sabe qué hacer, usted deberá corregir el programa para que esto no suceda.

Escriba programas para ejecutar las siguientes tareas:

(a) Visualizar cualquier tabla de multiplicar que pida el usuario.

(b) Preguntar al usuario que escriba su apellido, y luego el nombre de pila de su hijo, para luego mostrar el nombre completo de éste. (Nombre y apellido del hijo).



Las soluciones se dan en el Apéndice B, pero no deben consultarse hasta crear los programas o agotar las posibilidades de hacerlo.

| *Fin del Experimento 10.2* | |
|---|---|

El cuestionario para esta Unidad es UNI10CUEST.

# UNIDAD: 11


| | | |
|---|---|---|
| EXPERIMENTO 11.1 | PAG. | 79 |
| EXPERIMENTO 11.2 | | 83 |
| EXPERIMENTO 11.3 | | 89 |

Una de las características más interesantes de la programación es su riqueza y variedad. El mismo ordenador puede programarse como calculadora, máquina didáctica, instrumento musical, monitor de la evolución clínica de un paciente en un hospital o casi cualquier otra aplicación deseada. Esta versatilidad deriva del gran número de combinaciones posibles de los comandos BASIC.

Hasta aquí hemos practicado los siguientes comandos incorporados en programas:

PRINT, LET, GOTO, IF, INPUT, STOP y POKE

Naturalmente, el número de comandos es mucho mayor, pero antes de incrementar su número ensayaremos diversas facetas de estas instrucciones que nos son ya conocidas.



El más flexible de todos los comandos es IF. En las anteriores unidades de estudio hemos visto su uso en el control de bucles, pero sus posibilidades son mucho más amplias. Por ejemplo, puede comprobar los datos u otra información suministrada por el usuario, a fin de orientar la ejecución del programa en uno u otro sentido según exijan los datos aportados.

# EXPERIMENTO 11·1

Imaginemos que usted está organizando una agencia matrimonial y que quiere crear un programa para asesorar en cuestiones de edad. Según la tradición, el hombre debería casarse con una mujer que tiene la mitad de su edad, más siete. Ello implica a su vez que las mujeres deberían buscar un hombre que tuviera el doble de su edad menos 14.

Evidentemente, la primera acción del programa de asesoramiento matrimonial será preguntar al cliente la edad que tiene. Luego deberá determinar si se trata de un hombre o de una mujer. Se usará un mismo programa para hombres y mujeres, incluyendo un grupo separado de instrucciones para el asesoramiento de cada sexo. Finalmente, se incorporará un comando IF para escoger el grupo que corresponda en cada caso.

He aquí un programa idóneo para este fin. Estúdielo detenidamente y trate de ver la utilidad de cada comando:

```
10 INPUT"CUANTOS AÑOS TIENE";AG
20 INPUT"VARON O HEMBRA";SX$
30 IF SX$ = "VARON"THEN 70
40 PRINT"BUSQUE"
50 PRINT"UN HOMBRE DE";2*AG-14
60 STOP
70 PRINT"BUSQUE"
80 PRINT"UNA MUJER DE";AG/2 + 7
90 STOP
```

Usted habrá deducido que la variable AG se usa para almacenar la edad del cliente y que SX$ corresponde al sexo. La condición SX$ = "VARON" es verdadera si el cliente responde VARON a la pregunta "VARON O HEMBRA"?". La expresión AG/2+7 es la forma en BASIC para decir "la mitad de la edad más siete", mientras que 2*AG—14 significa "el doble de la edad menos catorce".

Cuando haya estudiado el programa, compruebe el grado en que ha comprendido el mismo tratando de predecir lo más exactamente posible la respuesta que aparecerá (a) para un joven de 20 años y (b) una chica de 22. Para ello, use los dos recuadros siguientes. El primer recuadro está ya parcialmente lleno.

```
RUN

QUE EDAD TIENE? 20

VARON O HEMBRA? V

BUSQUE UNA MUJER DE EDAD
```

(a)

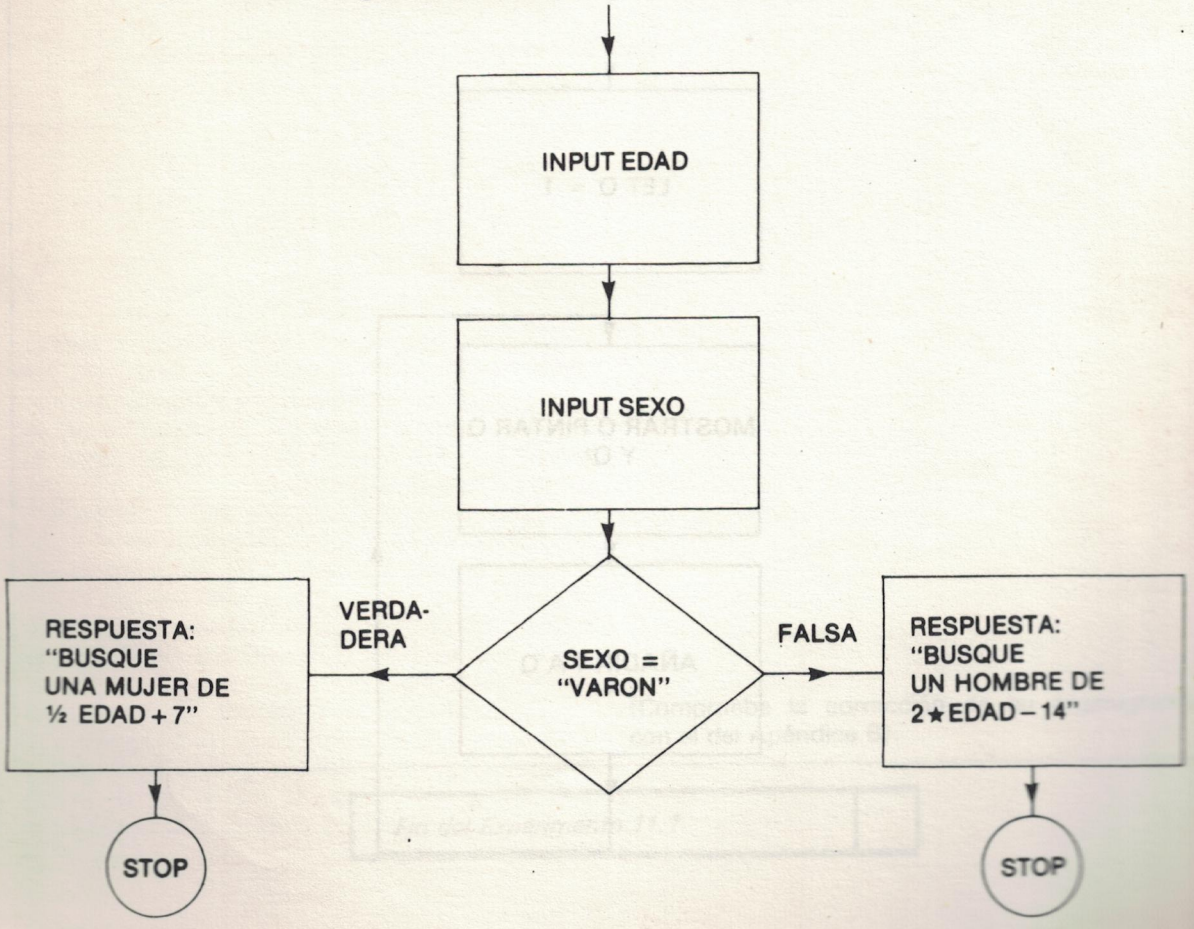
```
RUN
```

(b)

Ahora entre el programa en el C-64. Ejecútelo dando como datos de entrada los dos sexos y respectivas edades; compruebe si las respuestas coinciden con las que usted había previsto.

En este simple ejemplo se muestra como la acción del ordenador no debe ser fijada necesariamente de antemano por el programador, sino que puede depender de la información suministrada por el usuario.

Los programas a menudo tienen que realizar una serie de decisiones complejas, para cuya planificación se traza una especie de diagramas a los que se da el nombre de organigrama o "flow chart". El organigrama para el programa de asesoramiento descrito podría adoptar la siguiente modalidad:



Un organigrama está integrado por varios bloques, unidos por líneas que incluyen una flecha indicadora del sentido de ejecución del programa. Se usan cuatro tipos de bloques:

(a) Un *cuadrado* o *rectángulo*, para alojar la descripción de una acción simple, que luego puede representarse por una o dos instrucciones BASIC. En nuestro organigrama de muestra, los dos bloques superiores son ejemplos de este tipo. Las líneas con flecha señalan que se ejecuta la primera instrucción y luego la siguiente, en el orden indicado por la dirección de la flecha.

(b) Un *rombo* contiene una condición, que puede ser verdadera o falsa. El rombo tiene una línea de entrada, pero dos de salida, una etiqueta VERDADERA (o SI) y otra con el rótulo FALSA (o NO). El rombo correspondiente a una instrucción IF. Pida al ordenador que compruebe la condición y siga la línea VERDADERA o FALSA, según sea el resultado.

(c) El bloque *terminal*, constituido por un pequeño círculo con la palabra STOP y que marca el final del programa. Una "nube" (que no aparece en nuestro ejemplo). Señala una función que es demasiado compleja para describirla detenidamente en el mismo organigrama, por lo que suele desarrollarse en otra hoja.

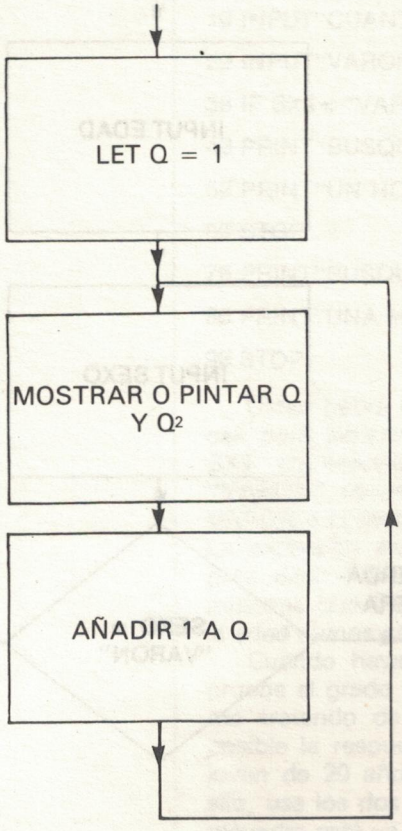
Un organigrama es en realidad el "mapa" de un programa, con flechas indicativas del camino a segur; de su observación se deduce:

- Un ordenador puede realizar sólo una tarea a un mismo tiempo (no varias)
- El orden de ejecución de las tareas es determinado por el programa.

Hay un comando que no se indica en ningún recuadro o bloque: GOTO. La razón de ello es que GOTO no incluye ninguna acción, sino que simplemente afecta el orden de ejecución de las instrucciones, por lo que se representa por una línea indicativa del bucle. Por ejemplo:

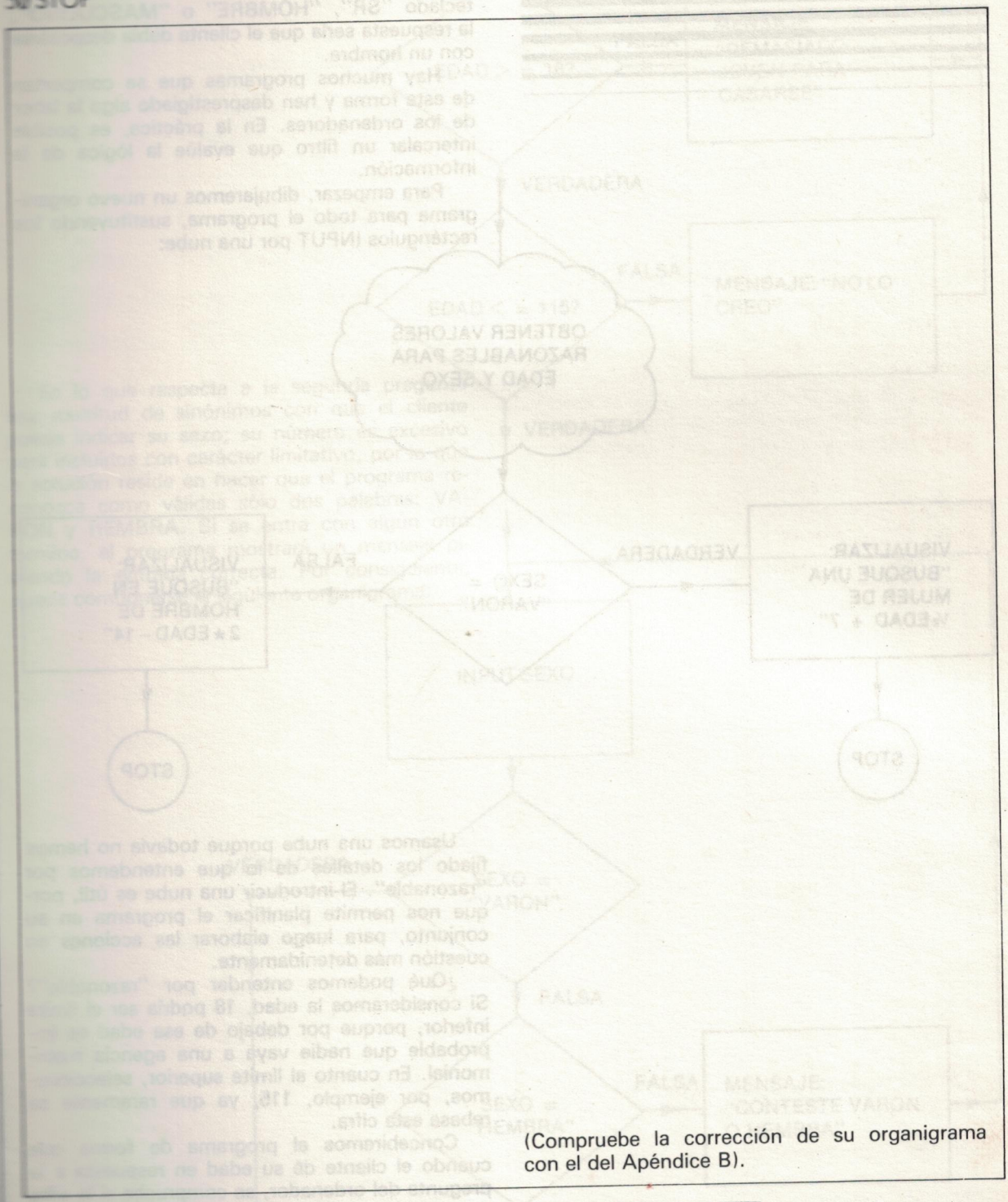
```
1Ø Q = 1
2Ø PRINT Q;Q★Q
3Ø Q = Q + 1
4Ø GOTO 2Ø
```

Tiene el caracter...



Dibuje un organigrama para el siguiente programa.

```
10 S=1
20 PRINT S,12*S
30 S=S+1
40 IF S < 13 THEN 20
50 STOP
```



(Compruebe la corrección de su organigrama con el del Apéndice B).

*Fin del Experimento 11.1*

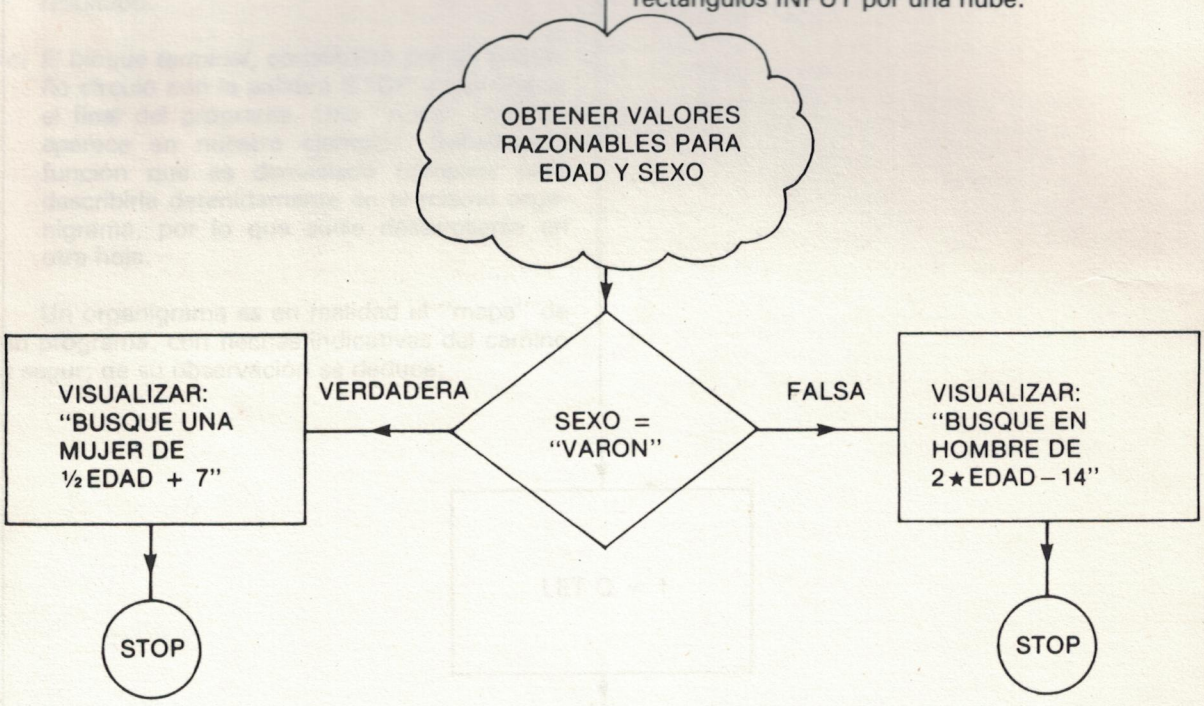
# EXPERIMENTO 11·2



Analicemos algunas facetas de la operación de los programas. En el programa creado para la agencia matrimonial, si se introducen datos incorrectos las respuestas serán absurdas (esta peculiaridad de los programas se define en la nomenclatura inglesa de la informática "gigo", las iniciales de "garbage in, garbage out"). Por ejemplo, si una chica manifestara que su edad era 6 años, la respuesta sería que su marido en perspectiva debería tener −2 años. De igual modo, si el usuario entrara alguna palabra que no fuera HOMBRE, el ordenador concluiría que era del sexo femenino; así, si se escribiera en el teclado "SR", "HOMBRE" o "MASCULINO", la respuesta sería que el cliente debía desposarse con un hombre.

Hay muchos programas que se comportan de esta forma y han desprestigiado algo la labor de los ordenadores. En la práctica, es posible intercalar un filtro que evalúe la lógica de la información.

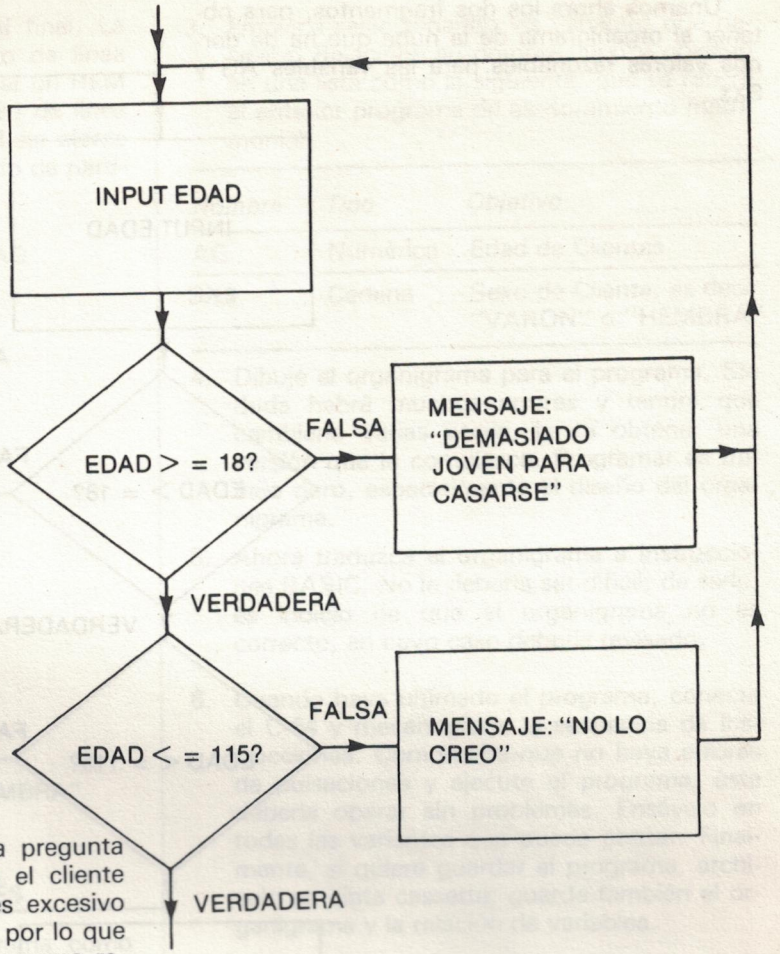
Para empezar, dibujaremos un nuevo organigrama para todo el programa, sustituyendo los rectángulos INPUT por una nube:



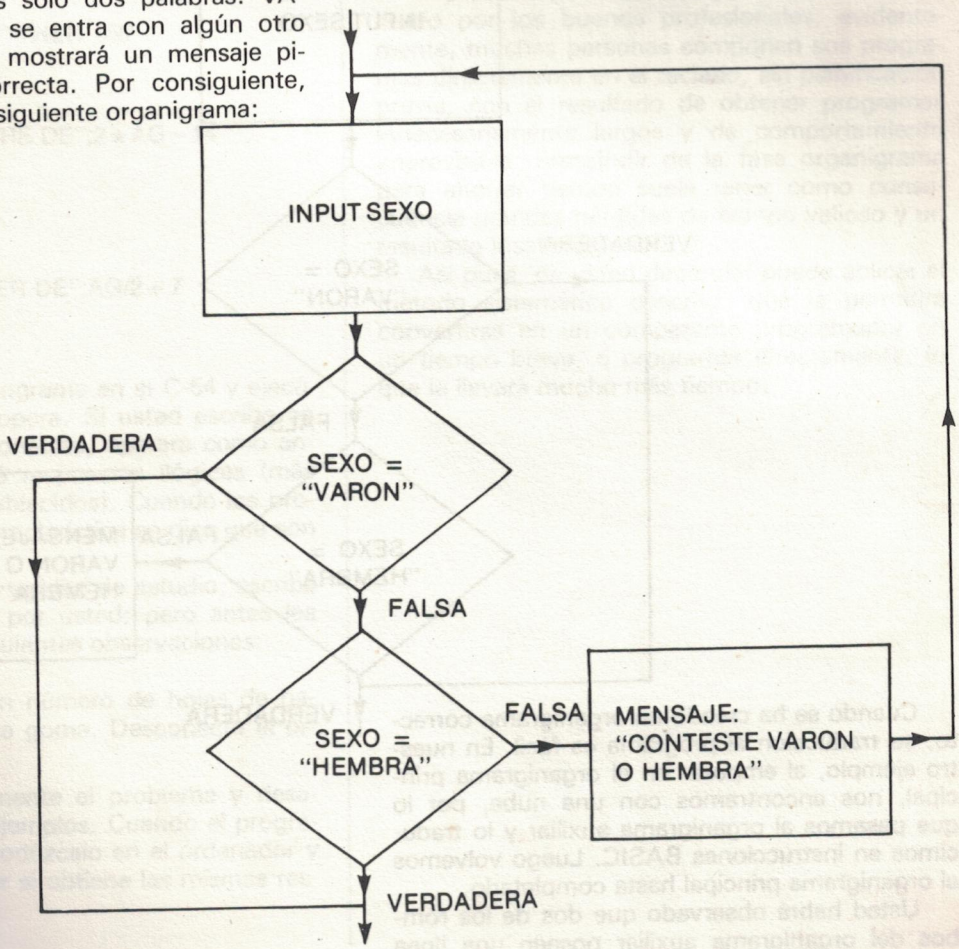

Usamos una nube porque todavía no hemos fijado los detalles de lo que entendemos por "razonable". El introducir una nube es útil, porque nos permite planificar el programa en su conjunto, para luego elaborar las acciones en cuestión más detenidamente.

¿Qué podemos entender por "razonable"? Si consideramos la edad, 18 podría ser el límite inferior, porque por debajo de esa edad es improbable que nadie vaya a una agencia matrimonial. En cuanto al límite superior, seleccionemos, por ejemplo, 115, ya que raramente se rebasa esta cifra.

Concebiremos el programa de forma que cuando el cliente dé su edad en respuesta a la pregunta del ordenador, se compruebe si la cifra entrada es razonable. Si no lo es, se pedirá al cliente que sea más realista. Vea el siguiente organigrama.

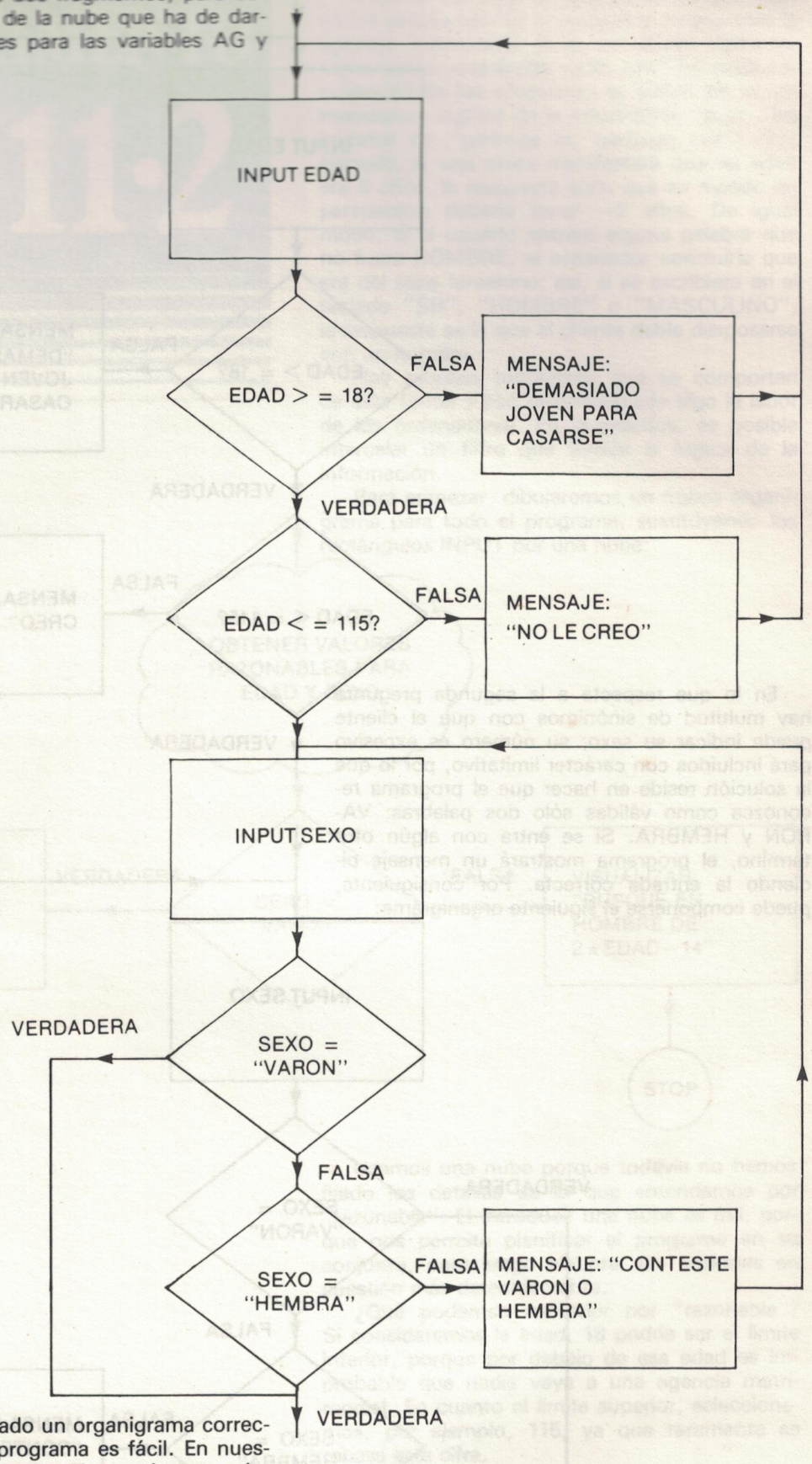



En lo que respecta a la segunda pregunta hay multitud de sinónimos con que el cliente puede indicar su sexo; su número es excesivo para incluirlos con carácter limitativo, por lo que la solución reside en hacer que el programa reconozca como válidas sólo dos palabras: VARON y HEMBRA. Si se entra con algún otro término, el programa mostrará un mensaje pidiendo la entrada correcta. Por consiguiente, puede componerse el siguiente organigrama:



Unamos ahora los dos fragmentos, para obtener el organigrama de la nube que ha de darnos valores razonables para las variables AG y SXS.






Cuando se ha creado un organigrama correcto, su traducción al programa es fácil. En nuestro ejemplo, al empezar en el organigrama principal, nos encontramos con una nube, por lo que pasamos al organigrama auxiliar y lo traducimos en instrucciones BASIC. Luego volvemos al organigrama principal hasta completarlo.

Usted habrá observado que dos de los rombos del organigrama auxiliar poseen una línea

VERDADERA que va directamente al final. La forma más fácil de adjudicar número de línea a cada instrucción IF consiste en poner un REM al final de la nube y utilizar el número de línea que se le conceda. El comando REM no ejerce ninguna acción, pero sirve de útil punto de partida. Así, obtenemos:

```
10 INPUT"CUANTOS AÑOS TIENE";AG
20 IF AG >=18 THEN 50
30 PRINT"DEMASIADO JOVEN PARA
   CASARSE"
40 GOTO 10
50 IF AG<=115 THEN 80
60 PRINT"NO LE CREO"
70 GOTO 10
80 INPUT"VARON O HEMBRA";SX$
90 IF SX$="VARON" THEN 130
100 IF SX$="HEMBRA" THEN 130
110 PRINT"CONTESTE VARON O HEMBRA"
120 GOTO 80
130 REM AG Y SX$ SON RAZONABLES
```

Ello va seguido del resto del programa como antes, cambiando sólo sus números de línea:

```
140 IF SX$="VARON" THEN 180
150 PRINT"BUSQUE"
160 PRINT"UN HOMBRE DE";2*AG-14
170 STOP
180 PRINT"BUSQUE"
190 PRINT"UNA MUJER DE";AG/2+7
200 STOP
```

Introduzca este programa en el C-64 y ejecútelo, para ver como opera. Si usted escribe en el teclado edades razonables, operará como antes, pero no admitirá respuestas ilógicas (más allá de los límites establecidos). Cuando los programas tienen esta característica se dice que son "robustos".

Para terminar esta unidad de estudio, escriba un programa creado por usted; pero antes lea detenidamente las siguientes observaciones:

1. Procúrese un gran número de hojas de papel, un lápiz y una goma. Desconecte el ordenador.
2. Estudie detenidamente el problema y desarrolle uno o dos ejemplos. Cuando el programa esté listo, introdúzcalo en el ordenador y ejecútelo, para ver si obtiene las mismas respuestas.
3. Empiece seleccionando las variables que necesita. Relacione su nombre, tipo y objetivo en una lista como la siguiente, que se refiere al anterior programa de asesoramiento matrimonial:

| *Nombre* | *Tipo* | *Objetivo* |
|---|---|---|
| AG | Numérica | Edad de Clientes |
| SX$ | Cadena | Sexo de Cliente, es decir "VARON" o "HEMBRA" |

4. Dibuje el organigrama para el programa. Sin duda habrá muchos errores y tendrá que cambiarlo varias veces, hasta obtener una versión que le complazca. Programar es trabajo duro, especialmente el diseño del organigrama.
5. Ahora traduzca el organigrama a instrucciones BASIC. No le debería ser difícil; de serlo, es indicio de que el organigrama no es correcto, en cuyo caso debería revisarlo.
6. Cuando haya ultimado el programa, conecte el C-64 y mecanografíe la secuencia de instrucciones. Compruebe que no haya errores de pulsaciones y ejecute el programa; éste debería operar sin problemas. Ensáyelo en todas las variantes que pueda pensar. Finalmente, si quiere guardar el programa, archívelo en cinta cassette; guarde también el organigrama y la relación de variables.

El procedimiento de trabajo descrito es el utilizado por los buenos profesionales; evidentemente, muchas personas componen sus programas directamente en el teclado, sin planificación previa, con el resultado de obtener programas innecesariamente largos y de comportamiento imprevisible; prescindir de la fase organigrama para ahorrar tiempo suele tener como consecuencia grandes pérdidas de tiempo valioso y un resultado insatisfactorio.

Así pues, de usted depende: puede aplicar el método sistemático descrito, que le permitirá convertirse en un competente programador en un tiempo breve, o programar directamente, lo que le llevará mucho más tiempo.

Ahora, trace un organigrama, escriba el programa, introduzca el ordenador y ejecútelo para el siguiente problema:

En Ruritania la contribución urbana se impone sobre las siguientes bases:

| | |
|---|---|
| Por cada puerta: | 57 £ |
| Por cada ventana: | 12 £ |
| Por cada techo de juncos: | 38 £ |
| Por cada tejado de obra: | 94 £ |

Partiendo del supuesto de que la casa posea una u otra modalidad de tejado, escriba un programa que pregunte los referidos detalles imponibles y a partir de los datos suministrados dé el impuesto total a abonar. Por ejemplo, la respuesta correcta para una casa de techo de juncos con una puerta y dos ventanas sería $(38 + 57 + 2 \star 12) = 119$ £.

Escriba, pues, un programa que permita calcular la contribución de cada una de las siguientes casas ilustradas, suponiendo que todas las puertas y ventanas se hallan en la fachada.

*Fin del Experimento 11.2*

Compruebe la solución en el Apéndice B.

# EXPERIMENTO 11·3



Cargue en memoria y ejecute el programa UNI11PROG. Luego lístelo y a la vista de sus comandos trace un organigrama y la pertinente relación o glosario de sus variables.

*Fin del Experimento 11.3*

# UNIDAD: 12


| | | |
|---|---|---|
| EXPERIMENTO 12.1 | PAG. | 94 |
| EXPERIMENTO 12.2 | | 97 |

No es necesario examinar detenidamente un programa para detectar algún bucle. Los bucles son tan frecuentes e importantes, que el lenguaje BASIC incluye un método simplificado de escribir sus detalles esenciales.

Usted recordará que hay cuatro elementos vitales para el control de todo bucle:

- La selección de la variable de control.
- El valor inicial de la variable de control.
- El valor final de la variable de control.
- El incremento, de la variable en cada ciclo.

Todos esos elementos pueden incorporarse en un comando especial que utiliza la palabra clave FOR. La instrucción FOR va seguida de otra complementaria que usa la palabra clave NEXT para marcar el final del cuerpo del bucle.

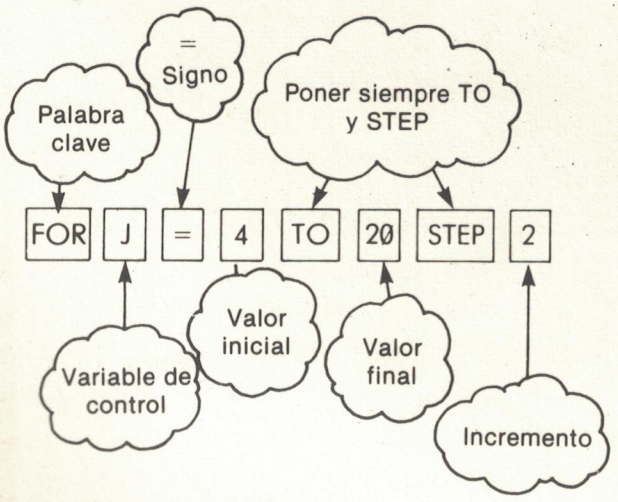
Compare los dos programas siguientes, que dan el mismo resultado:

```
1Ø J = 4                    1Ø FORJ = 4TO2ØSTEP2
2Ø PRINT J,J ★ 7            2Ø PRINT J,J ★ 7
3Ø J = J + 2                3Ø NEXT J
4Ø IFJ<22THEN2Ø
(Usando IF....THEN)         (Usando FOR....NEXT)
```

En los dos casos:

La variable de control es J
El valor inicial es 4
El valor final es 2Ø
El incremento es 2

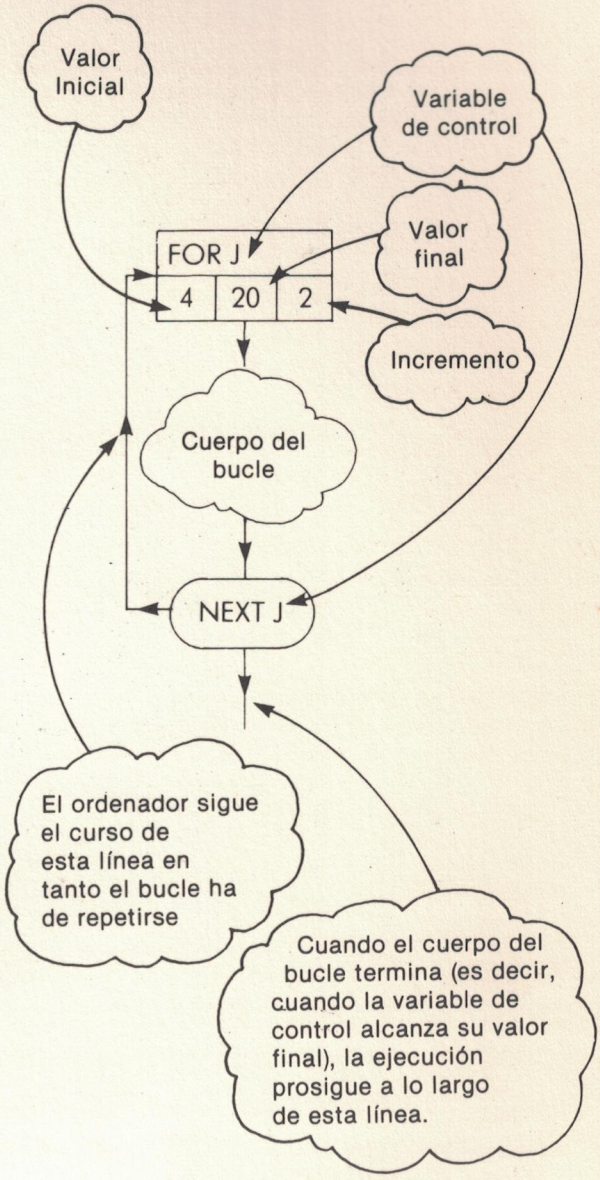
A continuación se explican los elementos que integran la línea 1Ø que contiene el comando FOR:




En el comando NEXT se cita el nombre de la variable de control, como ayuda para interpretar el programa.

Cada instrucción FOR posee su correspondiente NEXT, y entre ambas se halla contenido el cuerpo del bucle.

En los organigramas representamos los bucle con un recuadro especial subdividido, que no puede confundirse con ningún otro tipo de acción:



# EXPERIMENTO 12·1

Ahora traduzca el siguiente programa a notación FOR - NEXT. Para comprobar que está correcto, ejecute luego ambas versiones para asegurarse de que dan el mismo resultado:

(i)
```
1Ø FOR Q = 1 TO 16 STEP 5
2Ø PRINT Q;
3Ø NEXT Q
4Ø STOP
```

Traducción:

(ii)
```
1Ø FOR R = 38 TO 5Ø STEP 3
2Ø PRINT R;5Ø – R
3Ø NEXT R
4Ø STOP
```

Traducción:

Ahora traduzca el siguiente programa a notación FOR-NEXT. Para comprobar que está correcto, ejecute luego ambas versiones para asegurarse de que dan el mismo resultado:

```
1Ø PRINT"TABLA DE MULTIPLICAR POR NUEVE"
2Ø S = 1
3Ø PRINT S; "VECES 9 = ";
4Ø PRINT 9 ★ S
5Ø S = S + 1
6Ø IF S<13 THEN 3Ø
7Ø STOP
```

Traducción:

Tenga en cuenta las siguientes observaciones sobre los comandos FOR y NEXT:

(a) Si el incremento es 1, el elemento "STEP 1" al final de la instrucción FOR puede omitirse. El C-64 comprende de que se trata, e incrementará la variable de control de uno en uno.

(b) El bucle de control puede contar hacia atrás, usando un incremento negativo, o decremento. Por ejemplo, el programa.

```
1Ø FOR X = 1Ø TO 5 STEP – 1
2Ø PRINT X;
3Ø NEXT X;
```

exhibirá: 1Ø 9 8 7 6 5

en dicho orden.

(c) El cuerpo del bucle es ejecutado por lo menos una vez, aún cuando el valor final sea inferior al valor inicial y no se trata de un incremento negativo. Por ejemplo:

```
1Ø FOR R = 5 TO 3
2Ø PRINT R
3Ø NEXT R
```

exhibirá: 5

(d) Los valores del comando FOR no han de ser necesariamente números, sino que pueden consistir en expresiones que contengan otras variables. Por ejemplo, el siguiente programa indicará en la pantalla el número de símbolos "corazón" solicitado por el usuario. Ejecútelo y estúdielo detenidamente:



Dream commodore

```
10 INPUT"CUANTOS CORAZONES";H
20 FOR K = 1 TO H
30 PRINT" CTRL y RED ♥ ";
40 NEXT K
50 STOP
```

e) La variable de control no puede ser una cadena. Por ejemplo, la siguiente instrucción:

FOR X$ = "A" TO "ABBBB" STEP "B"

NO ES BASIC

daría un mensaje SYNTAX ERROR, porque es una construcción no válida. Utilizando sus conocimientos, prediga la solución que el ordenador dará al ejecutar los siguientes programas. Compare sus soluciones con las que da el C-64:

(i)
```
10 FOR A=1 TO 4
20 PRINT A★A;
30 NEXT A
40 STOP
```

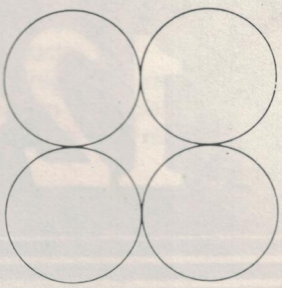
(ii)
```
10 FOR B=3 TO 0 STEP -1
20 PRINT B;
30 NEXT B
40 STOP
```

(iii)
```
10 FOR C = 5 TO 4
20 PRINT C;
30 NEXT C
40 STOP
```

(iv)
```
10 X=5
20 Y=9
30 Z=2
40 FOR W=X TO Y STEP Z
50 PRINT W;
60 NEXT W
70 STOP
```

Para facilitar la interpretación de los comandos FOR y NEXT hemos escogido cuerpos de bucle lo más simples posibles. Sin embargo, en la práctica pueden ser tan complejos como se desee, si bien hay que tener en cuenta que se ejecuta cada vez que el ordenador pasa por el bucle.

Supongamos que le pidan que construya una pirámide de base cuadrada utilizando bolas de cañón. Numeraremos las sucesivas capas 1, 2, 3, ..., empezando por arriba. La capa 1, al ser el vértice, necesitará sólo 1 bola. La capa 2 necesitará cuatro, dispuestas de la siguiente forma:



La capa 3 requerirá nueve, la capa 4, dieciséis, y así sucesivamente:

Está claro que el número de bolas de cañón que se necesita depende del número de capas que se quiera dar a la pirámide: una pirámide de tres capas requeriría 1+4+9, o sea 14, mientras que otra de cuatro capas precisaría 1+4+9+16, o sea 30.

Si la pirámide tiene que ser muy alta, vale la pena de escribir un programa, para que lo calcule el ordenador. El problema podría expresarse de la siguiente forma: "¿Cuántas bolas necesito para una pirámide de *tantas* capas?"

Al planificar el programa, un factor clave es el número de bolas que se necesita para cada capa. Si decimos que la capa 1 requiere 1, la capa 2 precisa 4, etc., vemos que el número corresponde en realidad al cuadrado de la capa; así, la número 7 nos dará 7★7, o 49 bolas de cañón.

Veamos ahora los detalles del programa. Pensemos ante todo en las variables que necesitaremos.

Consideremos las capas una a una; haremos que el ordenador calcule cuántas bolas entran en cada capa y sume dicho número en un total acumulativo. Al principio el total acumulativo será, naturalmente, cero, y al terminar nos dará la cifra necesaria para construir toda la pirámide, que será la solución al problema.

Seleccionemos como variable para el total acumulativo la variable RT.

Necesitamos otras dos variables:

(i) el número de capas de la pirámide; por ejemplo, L;

(ii) a medida que se ejecuta el programa, se calculará la capa 1, 2, etc.; necesitamos, pues, una variable que indique la capa que se está contabilizando; escojamos el nombre V. Puesto que la variable V irá adoptando todos los valores desde 1 a L —esta última el número indicativo de la capa inferior—, es obvio que constituye en realidad la variable de control en la instrucción FOR:

```
FOR V = 1 TO L

.....

NEXT V
```

*El glosario de las variables a utilizar en el programa es:*

| *Nombre* | *Objetivo* |
|---|---|
| RT | Para ir almacenando el total acumulativo |
| L | Número de capas en la pirámide |
| V | Número de la capa calculada en cada momento |

Ahora podemos anotar algunas de las acciones que debe ejecutar nuestro programa:

| | |
|---|---|
| Sumar al cuadrado de V a RT | (Añade el número de bolas de la capa V en cuestión al total acumulativo RT) |
| Print RT | (Visualiza el resultado) |
| Ajustar RT = Ø | (El valor inicial de RT es cero) |
| Input L | (Pregunta al usuario cuántas capas integran su pirámide) |
| FOR V = 1 TO L | (Control del bucle para evaluar cada capa). |
| STOP | |

Estos son los fragmentos de programa que necesitamos, pero hemos de ponerlos en el orden correcto. Ya sabemos que necesitamos un bucle, pero ¿dónde lo situamos? Los comandos pueden ir en las posiciones siguientes:

- antes de que el bucle empiece
- dentro del bucle (como parte del cuerpo del bucle)
- después de que haya terminado el bucle

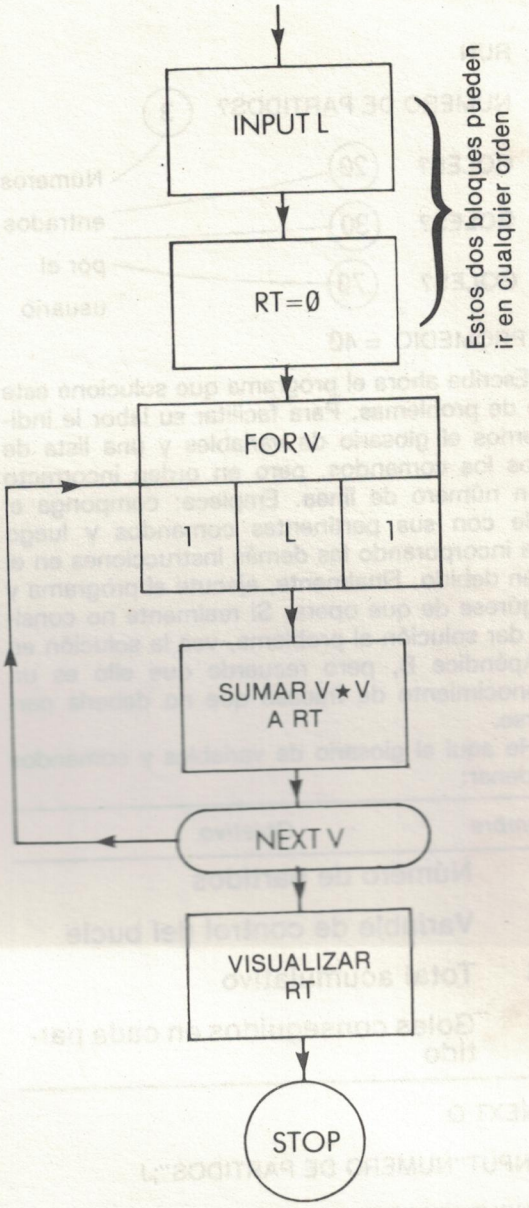
Tratemos de averiguar la situación requerida para cada comando.

El programa debe saber cuántas veces tiene que ejecutar el bucle antes de inicar dicha ejecución; por consiguiente, Input L debe preceder al bucle. Lo mismo es válido para la instrucción que pondrá RT a cero.

Deberán ir sumándose las sucesivas capas de bolas, y el comando para ir añadiendo cantidades a RT deberá repetirse varias veces, por lo que irá dentro del bucle.

A su vez, el comando PRINT habrá de situar necesariamente después del bucle, porque sólo entonces se conocerá el total acumulativo final, que es el valor que queremos averiguar.

**Dibujemos, pues, el pertinente organigrama:**




**El programa será, pues:**

```
1Ø INPUT"NUMERO DE CAPAS";L
2Ø RT=Ø
3Ø FOR V=1TO L
4Ø RT=RT+V★V
5Ø NEXT V
6Ø PRINT RT;"BOLAS REQUERIDAS"
7Ø STOP
```

**Entre este programa y ejecútelo.**

**Trate ahora de resolver el siguiente problema de programación. Se trata de calcular el promedio de goles conseguidos por un equipo de fútbol cualquiera. Naturalmente, se trata de sumar**

todos los goles realizados y dividirlos por el número de partidos. Escriba un programa que pregunte el número de partidos jugados y luego los goles obtenidos en cada partido. En la pantalla aparecería:

```
RUN
NUMERO DE PARTIDOS? 3
GOLES? 20
GOLES? 30
GOLES? 70
PROMEDIO = 40
```

Números entrados por el usuario



Escriba ahora el programa que solucione este tipo de problemas. Para facilitar su labor le indicaremos el glosario de variables y una lista de todos los comandos, pero en orden incorrecto y sin número de línea. Empiece: componga el bucle con sus pertinentes comandos y luego vaya incorporando las demás instrucciones en el orden debido. Finalmente, ejecute el programa y asegúrese de que opera. Si realmente no consigue dar solución al problema, vea la solución en el Apéndice B, pero recuerde que ello es un reconocimiento de fracaso que no debería permitirse.

He aquí el glosario de variables y comandos a ordenar:

| *Nombre* | *Objetivo* |
|---|---|
| J | Número de partidos |
| Q | Variable de control del bucle |
| RS | Total acumulativo |
| S | Goles conseguidos en cada partido |

```
NEXT Q
INPUT"NUMERO DE PARTIDOS";J
INPUT"GOLES";S
PRINT"PROMEDIO = ";RS/J
STOP
RS = 0
FOR Q = 1 TO J
RS = RS + S
```

*Fin del Experimento 12.1*

# EXPERIMENTO 12·2

Terminamos esta unidad de estudio con un problema que deberá solucionar sin ayuda. Imagine que va a comprar sellos a un estanco y que delante de usted hay alguien que nunca termina de comprar sus sellos, ya que el dependiente tiene que calcular el valor de cada lote y luego efectuar su suma.

Escriba un programa para ayudar al dependiente. Los mensajes de la pantalla deberían ser más o menos así:

```
RUN
NUMERO DE LOTES? 4
LOTE 1
  NUMERO DE SELLOS? 20
  VALOR UNITARIO? 15
LOTE 2
  NUMERO DE SELLOS? 5
  VALOR UNITARIO? 75
LOTE 3
  NUMERO DE SELLOS? 4
  VALOR UNITARIO? 1
LOTE 4
  NUMERO DE SELLOS? 100
  VALOR UNITARIO? 14
TOTAL = 2079 PTAS.
```

*Números entrados por el usuario.*

Número introducido por el usuario.

El programa no debe exceder de 10 comandos (incluyendo STOP). Cuatro de ellos formarán el cuerpo de un bucle, ejecútelo una vez para cada lote. Antes de empezar a escribir el programa, prepare un glosario de variables y dibuje un organigrama. Dedique tiempo a su solución.

Si, a pesar de todos sus esfuerzos, no consigue un programa satisfactorio, vuelva a revisar las últimas unidades de estudio, a partir de un punto en que se sienta convencido de haberlo entendido todo.

Finalmente, compare su programa con el del Apéndice B.

| *Fin del Experimento 12.2.* | |
|---|---|

El test (o quiz) para comprobar su dominio de los temas tratados en esta unidad de estudio se denomina "UNI12CUEST".

# UNIDAD: 13

EXPERIMENTO 13.1 PAG. 101

# EXPERIMENTO 13·1



Uno de los aspectos más sorprendentes del COMMODORE 64 es su gran versatilidad para generar una amplia gama de sonidos y notas musicales. El ordenador lleva incorporado un sintetizador, parecido al utilizado por las orquestas musicales modernas, con un generador de notas, filtros y ayudas especiales, que pueden ser controlados en programación BASIC.

El sintetizador es una unidad compleja. Su completa utilización representa un desafío, incluso para un profesional de la música. Afortunadamente Vd. puede sacarle un gran provecho con las nociones básicas que se estudian en esta unidad. Una discusión más amplia sobre el sintetizador se desarrolla en el segundo libro de esta serie.

Para tener una idea de las posibilidades de los sonidos que Vd. puede generar, cargue el programa "DEMOSON". Pruebe a seleccionar los sonidos. Asegúrese de tener ajustado adecuadamente el volumen de su televisor, ni muy alto, ni excesivamente bajo.

Si decide utilizar un sonido de los generados en un programa, es bastante sencillo. Una lista completa de los sonidos de demostración se detalla al final de esta unidad. No necesita más que copiar el grupo de sentencias que correspondan a los sonidos escogidos para su programa. Encontrará cada grupo de sonidos, incluido en apartados REM, como:

```
2000 REM  ★★★ LATIDO CARDIACO ★★★
.....
.....
.....
2120 REM  ★★★FIN DE LATIDO CARDIACO★★★
```

Comandos a copiar

Las sentencias a copiar son las existentes entre 2000 y 2150. No copie la sentencia de REM, o la de RETURN, que sigue al final de REM. Normalmente se copian las sentencias siguiendo el orden en el que están escritas. Si desea cambiar el orden tenga cuidado con el IF y GOTO: deben de cambiarse vigilando su coordinación en el programa. No necesita conocer con detalle las órdenes que copia, pero cópielas correctamente. Si se equivoca (especialmente en números largos, como 54272 o V en vez de VV), el programa no funcionará hasta que no haya subsanado el error. Si todo está correcto, ordene RUN.

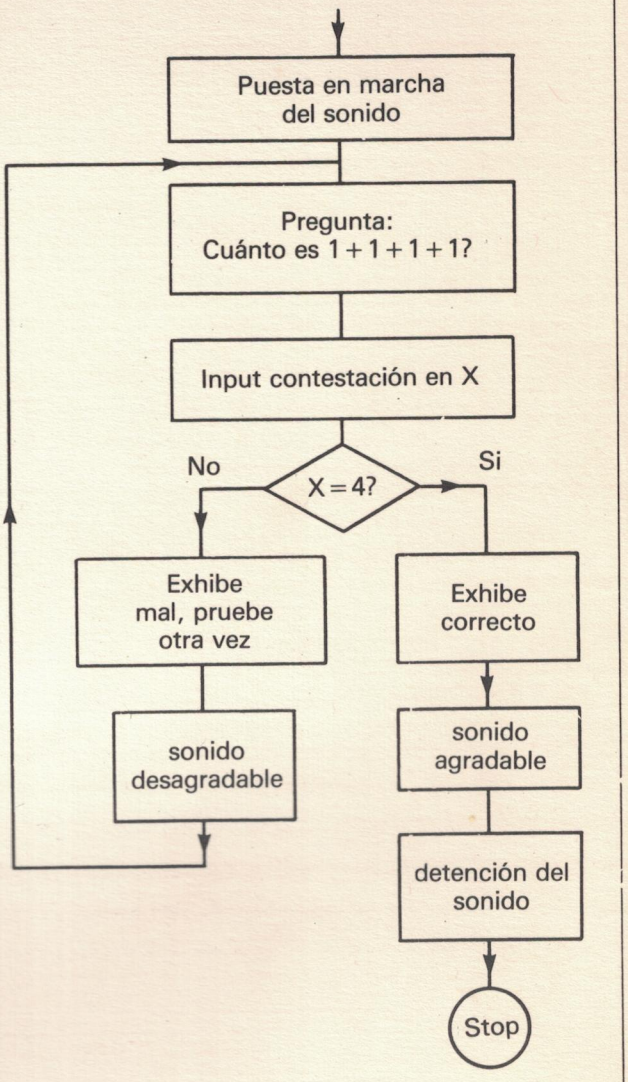
Un punto final: El ordenador genera sonidos, si previamente se ha ordenado la puesta en marcha del generador de sonidos. Se realiza mediante la sentencia:

```
POKE 54295,0:POKE 54296,15
```

Evidentemente este POKE debe de introducirse al inicio de cualquier programa que pretenda utilizar el sintetizador. Al finalizar el programa, el generador de sonido debe de apagarse, lo cual se realiza mediante

```
POKE 54296,0
```

Veamos un ejemplo: Diseñemos el programa para contestar a una pregunta. Si se escoge la respuesta correcta, el ordenador genera un risueño "piar" y exhibe en la pantalla un mensaje adecuado. Si al contrario se escoge una respuesta errónea, se produce un sonido desagradable, invitando el ordenador a que se responda de nuevo. El organigrama sería:



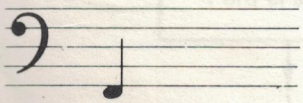
y el programa correspondiente sería:

```
10 POKE 54295,0:POKE 54296,15

        (Puesta en marcha del sonido)

20 PRINT "CUANTO ES 1+1+1+1";
30 INPUT X
40 IF X<>4 THEM 3080
50 PRINT "CORRECTO"

3010 VV=54272
3020 POKE VV+6,8: POKE VV+5,31
3030 POKE VV+1,180: POKE VV+4,33
3040 FORMM=1 TO 100: NEXTMM
3050 POKE VV+4,0

    (Comandos del sonido "piar" copiados del
            programa "DEMOSON")

3060 POKE 54296,0 (Detener sonido)

3070 STOP

        (Fin del programa)

3080 PRINT "MAL, PRUEBE OTRA VEZ"
4010 VV=54272: POKE VV+6,240
4020 POKE VV+1,4: POKE VV+5,0: POKE
     VV+4,33
4030 FOR NN=1024TO512STEP-8
4030 POKE VV+1, NN/256: POKE VV, NN and
     255
4050 NEXTNN
4060 POKE VV+4,0

  (Comandos para hacer un sonido desagradable
       como se hace en "DEMOSON)

4070 GOTO 20
```

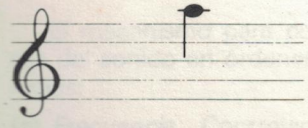
Bien, ahora veamos las posibilidades de generar sonidos. Seremos algo más técnicos. Si no le interesa profundizar en este tema puede pasar con toda tranquilidad a la siguiente unidad.

Empecemos con las notas musicales. Cada nota tiene cinco cualidades básicas o ATRIBUTOS y que diferencian una nota de la otra:

(a) TONO. Algunas notas son altas y otras bajas. En una partitura el tono usualmente se indica por la posición que ocupa en el pentagrama.



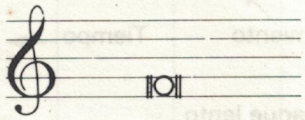
*Una nota baja*



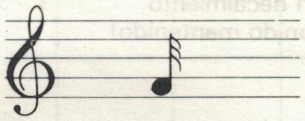
*Una nota alta*

El tono está relacionado directamente con la frecuencia. Es decir el número de vibraciones que se producen en un segundo. El LA medio tiene una frecuencia de 440 vibraciones (ciclos) por segundo; la frecuencia se multiplica por dos en cada octava aumentada (y disminuye a la mitad por cada octava que se desciende).

(b) DURACION. Algunas notas tienen una duración más larga que otras. En la partitura musical la duración de las notas viene indicada por un símbolo:



*Una nota larga*



*Una nota corta*

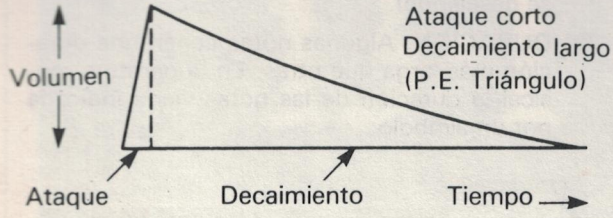
En la práctica el símbolo utilizado, no siempre indica la misma duración y sería más conveniente medir la duración de una nota en MILISEGUNDOS (unidades de 1/1000 segs.).

(c) Algunas notas musicales tienen SOSTENIMIENTO. Es decir mantienen un mismo grado de sonoridad mientras vibran. Los instrumentos de viento y los de cuerda generalmente producen notas con sostenimiento, otros sonidos, tales como una palmada o un golpe, se denominan TRANSITORIOS. Estos sonidos se inician con un grado de sonoridad que desciende rápidamente hasta dejar de ser percibidos. El piano es un excelente ejemplo de notas transitorias. Si se pulsa una tecla del piano, el sonido generado desciende su sonoridad gradualmente, pese a mantener la tecla pulsada.

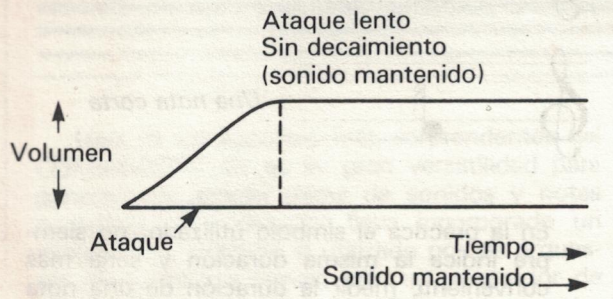
(d) Cada nota tiene un nivel de ATAQUE, que indica la rapidez con la que se alcanza la máxima sonoridad. Si hacemos vibrar una cuerda con algún objeto, las notas generadas alcanzan su máxima sonoridad casi instantáneamente, al igual que si golpeásemos un instrumento de percusión con una pieza metálica. Si la cuerda la hacemos vibrar con los dedos, el sonido obtenido es diferente: el TIEMPO DE ATAQUE ha cambiado y aunque existan tan sólo diferencias de fracciones de segundo el oido es capaz de percibirlo.

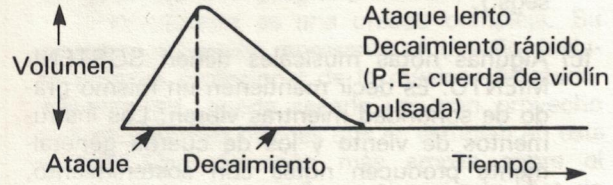
(e) Las notas transitorias presentan un DECAIMIENTO, que es una medida de pérdida de sonoridad, una vez que ésta se ha establecido. El DECAIMIENTO es más largo que el ataque y usualmente se mide en segundos. Un DECAIMIENTO corto hace que una nota se apague rápidamente, mientras que un DECAIMIENTO largo produce una sonoridad semejante a la causada por el claro tintineo de una campanilla.

La información sobre el nivel de SOSTENIMIENTO, ATAQUE y velocidad de DECAIMIENTO, se muestran a menudo en un gráfico llamado "ENVOLVENTE". Es un término técnico e indica la variación de sonoridad de una nota en función del tiempo. Tres envolventes típicas serían:






(f) Finalmente cada nota tiene un TIMBRE o calidad de sonido. Suave como el arpa o estridente como la trompeta.
En el curso de estas líneas hemos mencionado ocho nombres técnicos: TONO, FRECUENCIA, DURACION, NIVEL DE SOSTENIMIENTO, ATAQUE, DECAIMIENTO, ENVOLVENTE y TIMBRE. Debe perdonarnos por la aridez de la materia. El programa GENSON (generador de sonidos de prueba) está diseñado para ayudarle a comprender estas ideas, y evidentemente poder oir por Vd. mismo los cambios en la frecuencia, envolvente, etc... y como sus variaciones afectan al sonido. Cargue el programa y haga RUN. Pulse el espaciador tantas veces como desee. La nota que Vd. oirá, tiene las características que se exhiben en la pantalla.
- Frecuencia: 512 ciclos por segundo.
- Duración: 250 milisegundos.
- Nivel de sostenimiento: 0.
- Valor de ataque 0 (el ataque es instantáneo).
- Valor de decaimiento: 9 (alrededor de 1/4 seg.).
- Forma de onda: Triangular.

Ahora puede probar a cambiar los atributos de la nota, uno a uno y escuche lo que sucede. En primer lugar cambie la frecuencia, pulsando F, el número 1024 y RETURN; notará el ascenso en una octava en el tono de la nota. Cambie, ahora la frecuencia a 700, el tono disminuye. Puede variar el tono entre 20 y 40000 vibraciones por segundo.

A continuación, puede cambiar la duración. El programa acepta cualquier valor entre 1 y 5000 milisegundos. Si la duración es menor que el tiempo de caída la envolvente se acortará y la calidad de sonido de la nota no será la adecuada. Pruebe a realizarlo; reduzca la duración de la nota a 50 milisegundos.

El nivel de sostenimiento tan sólo tiene dos valores útiles; el 0 para una nota con caida y el 15 para una nota con sostenimiento. Pruebe ambos valores.

El ataque trabaja dé forma diferente: cuanto mayor es el número, menor es el ataque. Valores de 0 a 2 producen un sonido parecido al obtenido con instrumentos musicales; con valores de ataque altos se alcanza rápidamente el máximo grado de sonoridad. El efecto alcanzado es diferente; el sonido es más electrónico. Pruebe A = 6.

Escoja, ahora, valores de caída, comprendidos entre 6 (el sonido se apaga) y 11. Cuanto mayor sea el tiempo de caída, mayor será la duración de la sonoridad de la nota (que se irá silenciando lentamente).

Finalmente veamos los cuatro timbres (llamados a menudo "formas de onda"):

- "Triangular" (puede obtenerse pulsando "T").
— timbre suave.

- Diente de sierra (pulse "S").
— Timbre estridente.

- "Pulso" (pulse "P").
— Timbre ronco que a su vez puede variarse mediante su anchura entre 8 y 999. La información acerca de la anchura de pulso actual aparece en cuanto se selecciona esta forma de onda.

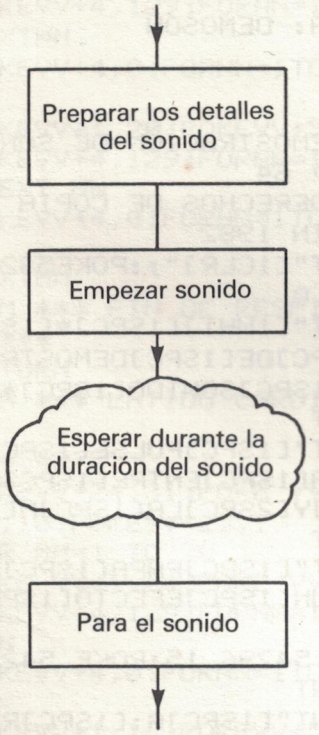
- "Ruido" (pulse "N"). Esta forma de onda no produce un sonido musical, se puede utilizar para simular explosiones, aviones, etc...

Le sugerimos que se tome algún tiempo en modificar los valores de los atributos de las notas, probando los resultados para familiarizarse con los diferentes efectos sonoros que se pueden obtener. Ahora, intente diseñar sus propios sonidos, utilizando la siguiente tabla (algunas líneas se desarrollan a modo de ejemplo):

| TIPO DE SONIDO | FRECUENCIA | DURACION | NIVEL DE SOSTE-NIMIENTO | VALOR DE ATAQUE | DECAI—MIENTO VALOR | FORMA DE ONDA | ANCHO DE PULSO |
|---|---|---|---|---|---|---|---|
| ARPA | 500 | 1000 | 0 | 1 | 11 | TRIANGULAR | — |
| SIRENA DE BARCO | 80 | 2500 | 15 | 7 | 0 | PULSO | 999 |
| EXPLOSION | 200 | 2000 | 0 | 0 | 12 | RUIDO | — |
| AVION | 4000 | 5000 | 0 | 10 | 13 | RUIDO | — |
| TROMPETA | 1000 | 500 | 15 | 3 | 0 | PULSO | 20 |
| OBOE | | | | | | | |
| PIZZICATO (CUERDA DE VIOLIN PULSADA) | | | | | | | |
| SILBATO DE POLICIA | | | | | | | |
| TRIANGULO | | | | | | | |
| TAMBOR | | | | | | | |
| TRIANGULAR | | | | | | | |
| PULSO | | | | | | | |
| RUIDO | | | | | | | |



Cuando Vd. diseñe un sonido especificando sus atributos, querrá evidentemente utilizarlo en el programa que haya escrito. El sintetizador está controlado por sentencias POKE, al igual que la estructura de la pantalla. El organigrama para generar un sonido es:




El mecanismo para convertir las características del sonido en órdenes, siguen las siguientes reglas:

(a) *Frecuencia*. Controlada por las direcciones de memoria 54272 y 54273. Si la frecuencia que se necesita (ciclos/segundo) es una variable llamada F, la secuencia de comandos será:

```
FF = 16 ★ F
POKE 54273, FF/265
POKE 54272, (FF-32768) AND 255
```

(b) El ancho del pulso (si lo necesita) está localizado en la posición 54275. Para un ancho del pulso de valor P, el comando adecuado es:

```
POKE 54275, P/8
```

(c) El nivel de sostenimiento se localiza en la posición 54278. Para sonidos fuertes utilizar:
POKE 54278,240
Para sonidos débiles, use: POKE 54278,Ø.

(d) Los niveles de ataque y decaimiento estan ambos controlados por la posición 54277. El valor es 16 veces el nivel de ataque más el nivel de decaimiento. Para valores A y D haga:
POKE 542277,16★A+D

Estos comandos preparan el sonido y puede utilizarlos en cualquier orden. El sonido se inicia por la posición 54276. Los valores siguientes controlan la forma de la onda sonora:

— Triangular POKE 54276,17
— Diente de sierra, POKE 54276,33
— Pulso, POKE 54276,65
— Ruido, POKE 54276,129

(e) Ahora el programa debe esperar durante la duración del sonido (T milisegundos). Esto se realiza fácilmente mediante un bucle FOR/NEXT, que no hace nada especial a excepción de contar:

```
FORC = 1 TO T
NEXT C
```

(f) Finalmente el sonido debe pararse colocando el valor Ø en la posición 54276:

```
POKE 54276,Ø
```

Utilicemos un ejemplo para aclarar estos detalles. Suponga que desea escribir un programa que en alguna zona genere el sonido de una sirena de barco. Los valores necesarios se encuentran en la segunda línea de la tabla de la página 106:

```
500 FF=16*80
510 POKE 54273,FF/256
520 POKE 54272, (FF-32768) AND255
530 POKE 54275,999/8
540 POKE 54278,240
550 POKE 54277,16*7+0
560 POKE 54276,65
570 FOR C=1 TO 2500
580 NEXT C
590 POKE 54276,0
```



Esto es:
Selecciona la frecuencia (líneas 500–520).
Selecciona el ancho del pulso (línea 530).
Establece el nivel de sostenimiento (línea 540)
Establece los valores de ataque y decaimiento (línea 550)
Empieza el sonido (línea 560)
Espera 2500 milisegundos (líneas 570–580)
Para el sonido (línea 590)

Los números de las líneas (500 a 590) son arbitrarios y cualquier otra secuencia de números de líneas, en el mismo orden, será igualmente válida. Acuérdese de poner en marcha el sonido al principio del programa.

Para finalizar esta unidad veamos algunos efectos sonoros. Dependen de la variación de la frecuencia en el generador, mientras se establece el sonido. El efecto no puede ser descrito en palabras, simplemente debe ser oido. Escriba y ejecute el siguiente programa:

```
10 POKE 54296,15
20 POKE 54278,240
30 POKE 54277,0
40 POKE 54276,33
50 FORN=10 TO80
60 POKE 54273,N
70 NEXTN
80 GOTO 50
```

Cuando desee parar el programa pulse RUN STOP y RESTORE a la vez.

Puede alterar la calidad del sonido cambiando los valores de los dos números de la línea 50, o bien utilizando el valor 17 ó 129, en vez del 33 en la sentencia 40.

Un sonido más complejo, puede obtenerse colocando el bucle de control dentro de otro bucle, de tal manera que los valores iniciales y finales cambien en función del tiempo:

```
10 POKE 54296,15
20 POKE 54278,240
30 POKE 54277,0
40 POKE 54276,33

50 FORC=1 TO 20
60 FORN=(40-C)TO(25-C)STEP-1
70 POKE 54273,N
80 NEXTN
90 NEXTC
100 GOTO 50
```

Se pueden cambiar los números de las expresiones entre paréntesis de la línea 60 (los paréntesis en esta línea se han incluido solo como indicación), o también que uno o ambos bucles vayan en dirección opuesta.

Esperamos que esta unidad haya despertado su interés en realizar cualquier tipo de sonidos con su COMMODORE 64. Esta área de programación es única, no tiene "respuestas correctas". El único fin perseguido es que los sonidos que Vd. produzca le sean agradables.

Para finalizar la unidad, ensaye a programar una sirena, o una bocina, que suba y baje el tono. Si tiene éxito, haga una copia de su programa para un uso futuro.

PROGRAMA: DEMOSON

```
0 REM DEMOSTRACION DE SONIDO (UN
IDAD 13) 64
10 REM DERECHOS DE COPIA (C) AND
REW COLIN 1982
20 PRINT"[1CLR]";:POKE53280,0:PO
KE53281,0
30 PRINT"[1WHT][1SPC]*[1SPC]PROG
RAMA[1SPC]DE[1SPC]DEMOSTRACION[1
SPC]DE[1SPC]SONIDO[1SPC]*"
40 PRINT
50 PRINT"[1SPC]PULSE[1SPC]UNA[1S
PC]LETRA[1SPC]ENTRE[1SPC]LA[1SPC
]A[1SPC]Y[2SPC]LA[1SPC]M[2SPC]"
60 PRINT
70 PRINT"[1SPC]PARA[1SPC]PRODUCI
R[1SPC]UN[1SPC]EFECTO[1SPC]SONOR
O"
80 POKE 54296,15:POKE 54295,0
100 PRINT
110 PRINT"[1SPC]A:[1SPC]RESPIRAC
ION[1SPC]PESADA"
120 PRINT"[1SPC]B:[1SPC]LATIDO[1
SPC]CARDIACO"
130 PRINT"[1SPC]C:[1SPC]PITIDO[1
SPC]"
140 PRINT"[1SPC]D:[1SPC]RUIDO[1S
PC]DE[1SPC]MOTOR"
150 PRINT"[1SPC]E:[1SPC]SILBIDO"
```

```
160 PRINT"[1SPC]F:[1SPC]VISITA[1
SPC]DE[1SPC]UN[1SPC]OVNI"
170 PRINT"[1SPC]G:[1SPC]EXPLOSIO
N"
180 PRINT"[1SPC]H:[1SPC]VOLAR[1S
PC]DE[1SPC]UNA[1SPC]MOSCA"
190 PRINT"[1SPC]I:[1SPC]SIRENA[1
SPC]DE[1SPC]POLICIA"
200 PRINT"[1SPC]J:[1SPC]MANDOLIN
A"
210 PRINT"[1SPC]K:[1SPC]VIOLIN"
220 PRINT"[1SPC]L:[1SPC]CANCION[
1SPC]"
230 PRINT"[1SPC]M:[1SPC]TRIANGUL
O"
300 GET A$: IF A$=""THEN 300
310 G=ASC(A$)-64
320 IF G<1 OR G>13 THEN 300
330 GG=1024+201+40*G
340 POKE GG,PEEK(GG)+128
350 ONGGOSUB1000,2000,3000,4000,
5000,6000,7000,8000,9000,10000,1
1000,12000,13000
360 POKEGG,PEEK(GG) AND 127
370 GOTO300
1000 REM *** RESPIRACION PESADA
***
1010 VV=54272
1020 POKE VV+6,0
1030 FOR MM=1 TO 10
1040 POKE VV+1,24:POKE VV+5,138
1050 POKEVV+4,129:FORNN=1TO500-2
5*MM:NEXTNN
1060 POKEVV+4,0:FORNN=1TO100:NEX
T NN
1070 POKEVV+1,20:POKEVV+5,155
1080 POKEVV+4,129:FORNN=1TO1500-
75*MM:NEXT NN
1090 POKEVV+4,0:FORNN=1TO200:NEX
T NN
1100 NEXTMM
1110 REM *** FIN DE RESPIRACION
PESDAD ***
1120 RETURN
2000 REM *** LATIDO CARDIACO ***

2010 VV=54272
2020 POKEVV+24,31:POKE VV+6,0:PO
KEVV+22,10:POKEVV+23,1
2030 FOR MM=1 TO 20
2040 POKE VV+1,24:POKE VV+5,51
2050 POKEVV+4,129:FORNN=1TO50 -M
M:NEXTNN
2060 POKEVV+4,0:FORNN=1TO100:NEX
T NN
2070 POKEVV+1,16:POKEVV+5,21
2080 POKEVV+4,129:FORNN=1TO 400-
10*MM:NEXT NN
2090 POKEVV+4,0:FORNN=1TO200:NEX
T NN
2100 NEXTMM
2110 POKE VV+24,15:POKEVV+23,0
2120 REM *** FIN DE LATIDO CARDI
ACO ***
2130 RETURN
3000 REM *** PITIDO ***
3010 VV=54272
3020 POKEVV+6,0:POKEVV+5,31
3030 POKEVV+1,180:POKEVV+4,33
3040 FORNN=1TO100:NEXT NN
3050 POKEVV+4,0
3060 REM *** FIN DE PITIDO ***
3070 RETURN
4000 REM *** RUIDO DE MOTOR ***
4010 VV=54272:POKE VV+6,240
4020 POKEVV+1,4 :POKEVV+5,0:POKE
VV+4,33
4030 FORNN=1024 TO 512 STEP -8
4040 POKE VV+1,NN/256:POKE VV,NN
 AND 255
4050 NEXTNN
4060 POKEVV+4,0
4070 REM *** FIN DE RUIDO DE MOT
OR ***
4080 RETURN
5000 REM *** SILBIDO ***
5010 VV=54272
5020 POKE VV+6,0:POKE VV+5,9
5030 POKE VV+4,17
5040 FOR NN=80 TO 140STEP4:POKE
VV+1,NN:NEXTNN
5050 POKE VV+4,0:POKE VV+5,74
5060 FOR NN=1TO120:NEXTNN
5070 POKE VV+4,17
5080 FOR NN=100TO120:POKEVV+1,NN
:NEXT NN
5090 FOR NN=120TO60STEP-0.4:POKE
VV+1,NN:NEXT NN
5100 POKE VV+4,0
5110 REM *** FIN DE SILBIDO ***
5120 RETURN
6000 REM *** VISITA DE UN OVNI *
**
6010 VV=54272:POKE VV+1,40
6020 POKE VV+6,0:POKE VV+5,204:P
OKE VV+4,0:POKE VV+4,129
6030 FORNN=0TO2500:NEXT NN
6040 POKE VV+6,240:POKE VV+5,0:P
OKE VV+4,0
6050 POKE VV+4,17
6060 FOR MM=20 TO 6 STEP -1
6070 FOR NN=   MMTO50-MM
6080 POKEVV+1,NN
6090 NEXT NN,MM
6100 POKE VV+1,3:POKE VV+4,0
6110 FOR NN=1TO 10
6120 POKE VV+4,33:FOR MM=1TO50:N
EXTMM
6130 POKE VV+4,0:FOR MM=1TO200:N
EXTMM
6140 NEXTNN
6150 POKEVV+6,0:POKEVV+5,5:FOR N
```

```
N=3 TO 100 STEP 5
6160 POKEVV+1,NN:POKE VV+4,33
6170 FOR MM=1TO40-0.25*NN :NEXT
MM
6180 POKE VV+4,0:FOR MM=1 TO 120
-NN:NEXTMM
6190 NEXT NN
6200 POKE VV+5,31:POKE VV+4,129
6210 FOR NN=40TO200 STEP 0.1
6220 POKE VV+1,NN
6230 NEXT NN
6240 POKE VV+4,0
6250 REM *** FIN DE VISITA DE UN
 OVNI ***
6260 RETURN
7000 REM *** EXPLOSION ***
7010 VV=54272:POKE VV+6,0:POKE V
V+5,12
7020 POKE VV+1,20:POKE VV+4,129
7030 FORNN=1TO2500:NEXTNN
7040 POKE VV+4,0
7050 REM *** END OF EXPLOSION **
*
7060 RETURN
7070 POKE VV+4,0
7080 REM *** FIN DE EXPLOSION **
*
7090 RETURN
8000 REM *** VOLAR DE UNA MOSCA
***
8010 VV=54272
8020 POKE VV+6,0:POKE VV+5,205
8030 POKE VV+1,48+2*RND(0)
8040 POKE VV+4,33
8050 FORNN=1TO200:POKE VV+1,48+I
NT(2*RND(0))
8055 FORMM=1TO10:NEXTMM
8060 NEXTNN
8070 POKE VV+4,0
8080 REM *** FIN DE VUELO DE UNA
 MOSCA ***
8090 RETURN
9000 REM *** SIRENA DE POLICIA *
**
9010 VV=54272:POKEVV+6,0:POKEVV+
5,237
9020 POKEVV+2,200:POKEVV+3,0
9025 AA=55:BB=42
9030 POKEVV+1,60:POKEVV+4,65
9040 FOR NN=1TO12
9050 POKEVV+1,AA
9060 FORMM=1TO300:NEXTMM
9070 POKEVV+1,BB
9080 FORMM=1TO300:NEXTMM
9085 IF NN=6THEN AA=53:BB=40
9090 NEXT NN
9100 POKEVV+4,0
9110 REM *** FIN DE SIRENA DE PO
LICIA ***
9120 RETURN
10000 REM *** MANDOLINA ***
10010 VV=54272:POKE VV+6,0:POKE
VV+5,8
10020 PP=1000:QQ=2↑(1/12)
10030 FOR KK=0TO4
10040 FOR JJ=0TO 11
10050 IFJJ=1ORJJ=3ORJJ=6ORJJ=8OR
JJ=10THEN10080
10060 POKE VV+1,PP/256:POKEVV,PP
 AND 255
10070 POKEVV+4,0:POKEVV+4,33:FOR
NN=1TO 50:NEXTNN
10080 PP=PP*QQ
10090 IFKK=4THEN 10110
10100 NEXT JJ,KK
10110 FORNN=1TO200:NEXTNN
10120 POKE VV+4,0
10130 REM *** FIN DE MANDOLINA *
**
10140 RETURN
11000 REM *** VIOLIN ***
11010 VV=54272:POKEVV+6,32:POKEV
V+5,106
11020 PP=20000:QQ=2↑(1/12)
11030 POKEVV+3,50
11040 FOR KK=1TO3
11050 FOR JJ=12 TO 1 STEP -1
11060 IF JJ=1ORJJ=4ORJJ=6ORJJ=90
RJJ=10THEN 11150
11070 POKEVV+4,0: POKEVV+4,65
11080 FOR NN=1TO4
11090 ZZ=1.005*PP:POKEVV+1,ZZ/25
6:POKEVV,ZZ AND 255
11100 FORMM=1TO25:NEXTMM
11120 ZZ=0.995*PP:POKEVV+1,ZZ/25
6:POKEVV,ZZ AND255
11130 FORMM=1TO25:NEXTMM
11140 NEXT NN
11150 PP=PP/QQ
11160 IFKK=3THEN 11180
11170 NEXT JJ,KK
11180 FORMM=1TO500 :NEXTMM:POKEV
V+4,0
11190 REM *** FIN DE VIOLIN ***
11200 RETURN
12000 REM *** CANCION ***
12010 RESTORE
12020 VV=54272
12030 POKE VV+6,  0:POKEVV+5,64+
13
12040 POKE VV+3,99 :PP=2500
12050 READ WW: IF WW=0THEN POKE
54276,0:RETURN
12060 RR=INT(WW/10):QQ=(WW-10*RR
)*PP
12070 POKE 54273,QQ/256:POKE5427
2,QQ AND 255
12080 POKE54276,0:POKE 54276,65:
FORNN=1TO70 *(RR+1):NEXTNN
12090 GOTO 12050
12100 DATA 25,6,24,5,3,3,3,4,5,6
,25,6,24,5,3,3,3,64,3,3,5,4,4,5,
```

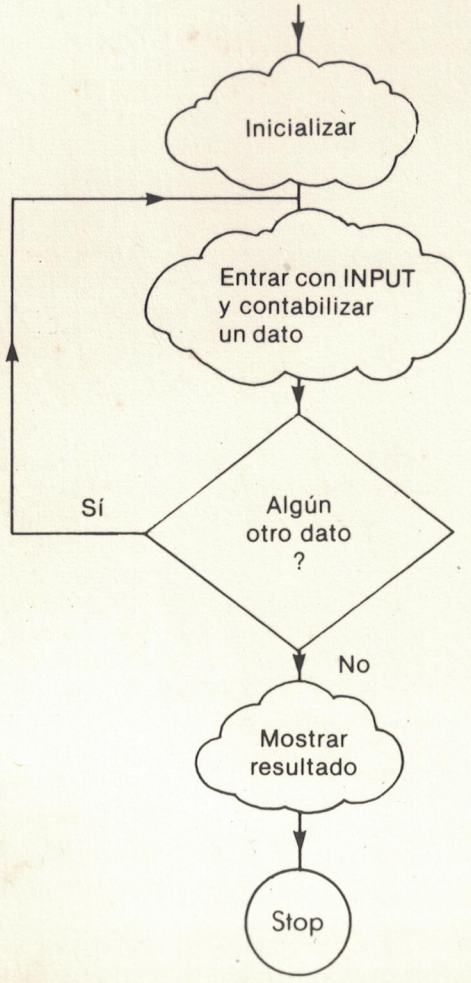
```
3,3,5,4,5,6
12110 DATA15,6,24,5,3,3,3,4,5,6,
25,6,24,5,3,3,3,94,0
12120 REM *** FIN DE CANCION ***

13000 REM *** TRIANGULO ***
13010 VV=54272
13020 POKE VV+6,0:POKEVV+5,12:PO
KEVV+1,200
13030 POKEVV+3,40
13040 FOR NN=1TO5
13050 POKE VV+4,0:POKE VV+4,65
13060 FORMM=1TO500:NEXTMM
13070 NEXTNN
13080 POKEVV+4,0
13090 REM *** FIN DE TRIANGULO *
**
13100 RETURN
```

# UNIDAD: 14


EXPERIMENTO 14.1 PAG. 114
EXPERIMENTO 14.2 117

En esta unidad de estudio trataremos un tipo de aplicación muy común: se suministrará a la máquina una serie de datos, que son contabilizados automáticamente para mostrar el resultado final. Por ejemplo, para comprobar su cuenta corriente puede escribir un programa que descuente automáticamente los talones que extienda y sume los ingresos que efectue, dando el balance al final de la semana. Otro ejemplo podría consistir en un programa que permitiera al usuario introducir todas las notas de una clase para que el maestro supiera su promedio.

Todos los programas de este tipo poseen una pauta similar, que podría reflejarse en el siguiente organigrama:




Un ejemplo muy simple es el siguiente programa, que introduce 10 números y calcula su promedio:

```
10 S=0   }
20 P=1   } Inicializar

30 INPUT X  }
40 S=S+X    } Leer y contabilizar un dato

50 P=P+1                  }
60 IF P<11 THEN 30        } ¿Algún dato más?

70 PRINT"PROMEDIO=";S/10  } Mostrar resultado

80 STOP
```

## Glosario

S: Se usa para sumar valores de datos sucesivos
P: Se usa para contar los datos sucesivos.
X: Se usa para introducir los datos sucesivos.

Si no comprende el funcionamiento de este programa, proceda a su rastreo con los valores introducidos con INPUT 3, 6, 2, 7, 0, 9, 8, 3, 12, 10.

En este ejemplo hemos usado una instrucción IF...THEN para controlar el bucle a fin de que su construcción fuera más sencilla; sin embargo, en la práctica usaríamos en su lugar el comando FOR...NEXT:

```
10 S=0

20 FOR P=1 TO 10

30 INPUT X

40 S=S+X

50 NEXT P

60 PRINT"PROMEDIO";S/10

70 STOP
```

Examinemos la parte del programa en que se pregunta si hay algún dato más. En nuestro ejemplo contestábamos a esa pregunta contando los datos sucesivos y empleando la condición P<11, que se mantenía verdadera hasta llegar al décimo dato que se introducía y sumaba al total acumulativo. Sin embargo, este método exige que el programador sepa de antemano cuántos datos van a introducirse, por lo que en la práctica carece de utilidad: para hallar el promedio de 11 cantidades requeriríamos otro programa, y lo mismo ocurriría con cualquier otro número de cantidades.

Es posible escribir un programa mucho más práctico, haciendo que sea el usuario quien especifique el número de datos a entrar:

```
1Ø S = Ø
2Ø INPUT"CUANTAS CANTIDADES";N
3Ø FOR P = 1 TO N
4Ø INPUT X
5Ø S = S + X
6Ø NEXT P
7Ø PRINT"PROMEDIO"; S/N
8Ø STOP
```

Vèa el uso de N en vez de 1Ø

## Glosario

S: Se usa para sumar el valor de los datos

P: Se usa para contar los datos

X: Se usa para entrar los datos

N: Se usa para retener el número de datos

Esta segunda versión del programa no aporta novedades, pues ya hemos realizado este tipo de programación en anteriores unidades de estudio; sin embargo, ¿qué pasa si el usuario quiere introducir una gran cantidad de datos (mil o más)? Sería poco práctico que tuviera que contarlos...

Para superar esa dificultad, en vez de controlar los ciclos del bucle especificando de antemano su número de ciclos, se construye de modo que el usuario puede indicar cuándo la corriente de datos ha terminado. Por ejemplo, el usuario puede contestar a la pregunta "algún otro dato?" con cada ciclo del bucle. Ello nos llevaría al siguiente tipo de programa:

```
1Ø S = Ø
2Ø N = Ø
3Ø INPUT"SIGUIENTE";X
4Ø S = S + X
5Ø N = N + 1
6Ø INPUT"ALGUN OTRO?";M$
7Ø IF M$ = "SI" THEN 3Ø
8Ø PRINT"PROMEDIO";S/N
9Ø STOP
```

## Glosario

S: Se usa para sumar los valores de los sucesivos datos

N: Se usa para contar los datos

X: Se usa para introducir los datos

M$: Se usa para contestar a la pregunta "Algún otro"?

Si usted ejecuta este programa, en la pantalla irá apareciendo:

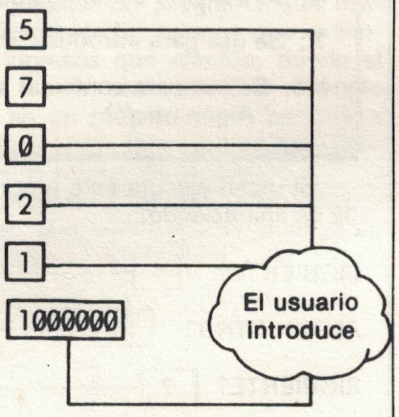
```
SIGUIENTE? 4
ALGUN OTRO? Sí
SIGUIENTE? 7
ALGUN OTRO? Sí
SIGUIENTE? 2Ø
ALGUN OTRO No
PROMEDIO 7
BREAK IN 90
READY
```

El usuario mecanografía

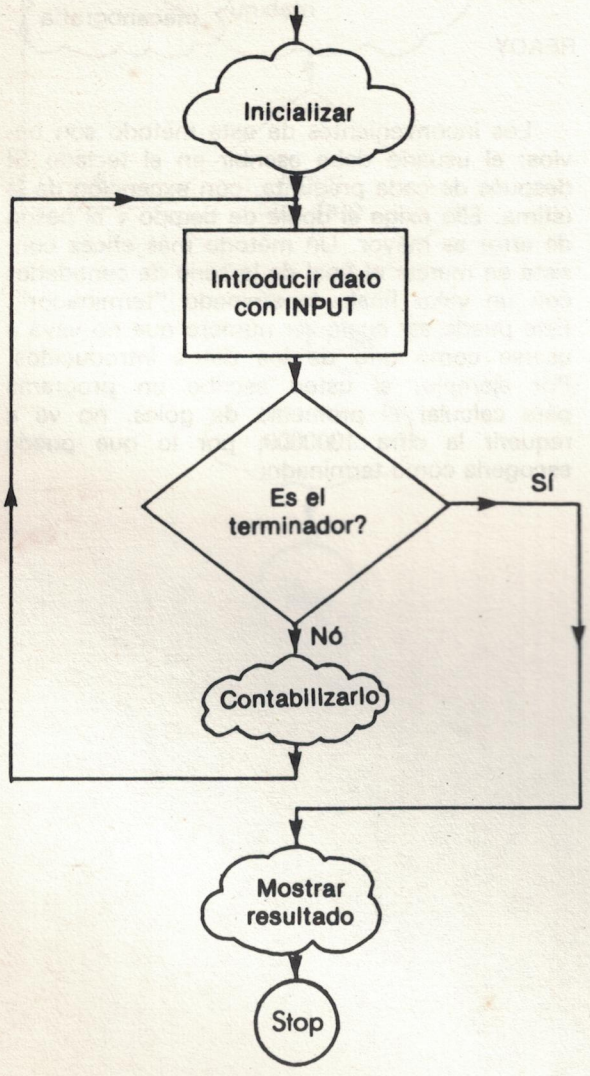
Los inconvenientes de este método son obvios: el usuario debe escribir en el teclado SI después de cada pregunta, con excepción de la última. Ello exige el doble de tiempo y el riesgo de error es mayor. Un método más eficaz consiste en marcar el final de la serie de cantidades con un valor final, denominado "terminador". Este puede ser cualquier número que no vaya a usarse como uno de los datos introducidos. Por ejemplo, si usted escribe un programa para calcular el promedio de goles, no va a requerir la cifra 1000000, por lo que puede escogerla como terminador.

Los mensajes mostrados por un programa de este tipo podrían ser:

```
USAR 1000000 PARA
TERMINAR INPUT
SIGUIENTE? 5
SIGUIENTE? 7
SIGUIENTE? 0
SIGUIENTE? 2
SIGUIENTE? 1
SIGUIENTE? 1000000
PROMEDIO 3
BREAK...
```




Para usar este sistema hemos de modificar todo el organigrama; especialmente la pregunta "algún otro"? debe anteponerse al bloque que contabilizaba cada dato, ya que de lo contrario el valor del terminador sería tratado como una cantidad más. Y obtendríamos un resultado erróneo.




El correspondiente programa para encontrar el promedio es como sigue:

```
10 PRINT"USAR 1000000 PARA"
20 PRINT"TERMINAR INPUT"
30 S=0
40 N=0
50 INPUT"SIGUIENTE";X
60 IF X = 1000000 THEN 100
70 S=S+X
80 N=N+1
90 GOTO 50
100 PRINT"PROMEDIO=";S/N
110 STOP
```

## Glosario

S: Se usa para sumar los valores sucesivos

N: Se usa para contar el número de valores

X: Se usa para introducir cada valor

Hemos visto las siguientes cuatro formas de indicar cuantos valores van a ser introducidos, en el programa.

1. El número de datos es especificado por el programador. Usado por principiantes, sin utilidad práctica.

2. El número de datos es especificado de antemano por el usuario. Un método idóneo, si el número de datos no excede de 20.

3. El usuario indica después de cada dato si le sigue algún otro. Innecesariamente tedioso.

4. La serie de datos termina con un valor final determinado. Es un buen método, mejor que los otros.

# EXPERIMENTO 14·1

Escriba un sencillo programa que permita introducir el saldo anterior de su cuenta corriente y que pida, el valor de los sucesivos talones que usted ha extendido, para que al terminar le muestre el saldo final. Puede usar el número cero como terminador, ya que no va a expedir talones por dicho valor y los cancelados no necesitan contabilización. Prescinda por ahora de los ingresos que efectúe en su cuenta. Desarrolle el programa de modo que en la pantalla aparezca una de las dos versiones siguientes:

(a) SALDO ANTERIOR? 524

ESCRIBA DETALLES DE

CHEQUES, USAR Ø PARA TERMINAR

CANTIDAD? 1.730

CANTIDAD? 2.000

CANTIDAD? Ø

A escribir por el usuario

SU DEUDA ES 3.206

(b) SALDO ANTERIOR? 4.020

ESCRIBA DETALLES DE

CHEQUES, USAR Ø PARA TERMINAR

CANTIDAD? 3.500

CANTIDAD? 1.500

CANTIDAD? 3.000

CANTIDAD? Ø

A escribir por el usuario

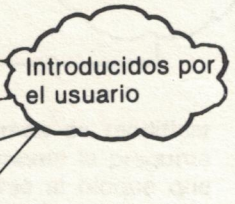
SU DEUDA ES

El valor de la variable usada para almacenar el saldo actual, será negativo si se extienden talones en exceso de la cuenta. La correcta condición para comprobar si éste es el caso es, por ejemplo B<Ø (si selecciona esta variable para contener el saldo). Dibuje un organigrama, glosario de variables y el programa. Luego compruébelo con el expuesto en el Apéndice B.

| *Fin del Experimento 14.1* | |
|---|---|

En la mayoría de los programas concurren datos de diferente tipo, que exigen un tratamiento diverso. A este fin, en el interior del bucle principal se incorporan comandos IF. Supongamos que usted está jugando dinero con una moneda a cara o cruz, y no para de perder, y se da cuenta de que sale más veces cruz que cara. Naturalmente, puede solventar el misterio arrojando varias veces la moneda y contando las veces que sale un lado u otro. Supongamos que pide ayuda al C-64 para que contabilice ambos resultados. El programa mostraría en pantalla lo siguiente:

```
ESCRIBA H SI SALE CARA
T SI SALE CRUZ
E PARA TERMINAR
SIGUIENTE? H
SIGUIENTE? H
SIGUIENTE? T
.....y así sucesivamente 547 líneas
SIGUIENTE? E
DE 547 TIRADAS
HUBO 490 CARA
Y 57 CRUZ
READY
```




Dibujemos el pertinente organigrama y preparemos el glosario de variables.

En la anterior planificación de los mensajes que deseamos en la pantalla hemos seleccionado un valor especial, E, para terminar la serie de datos. El organigrama será idéntico al de la pág. 109, ya que lo único que se precisa es extender las nubes.

Es obvio que necesitamos tres variables:

| |
|---|
| H: Para contar el número de caras |
| T: Para contar el número de cruces |
| I : Para introducir cada dato |

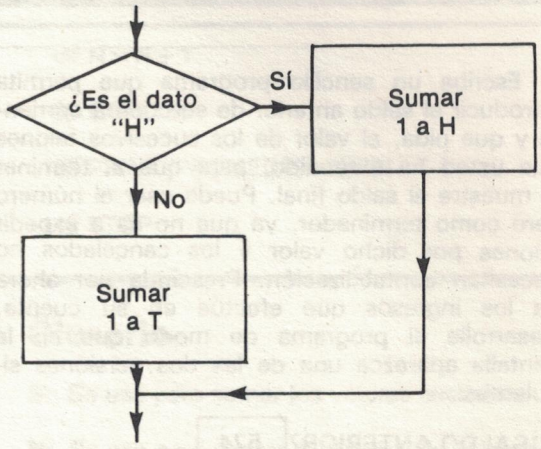
(como de costumbre, escogemos H y T al azar como letras para representar variables).

Alguien podría pensar que cabía incluir una cuarta variable, para contar el número de tiradas, pero no hay necesidad de ello: este total puede obtenerse con la expresión H + T.

Empecemos con la sección de iniciación del programa. Hemos de hacer dos cosas:

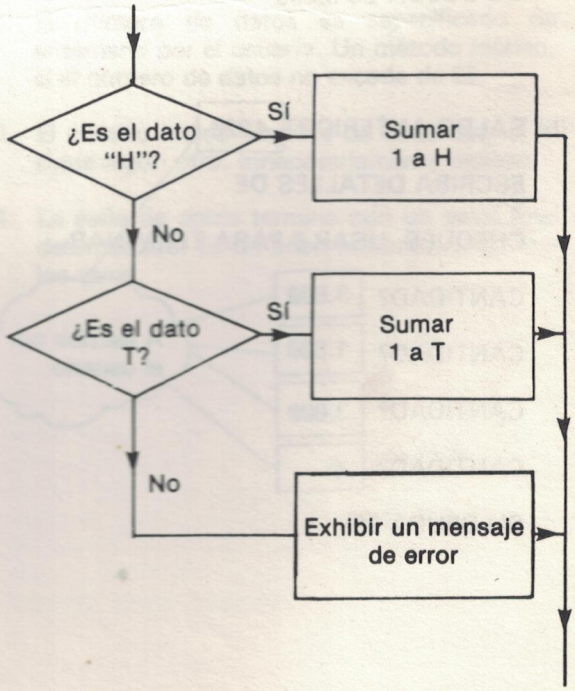
- Ajustar las variables H y T a cero.
- Seleccionar mensajes

Elaboremos ahora la nube que hay dentro del bucle principal, que contabiliza cada nuevo dato. En esta fase ya se habrá filtrado, o excluido, la E, por lo que cada dato será lógicamente H o T. La tarea básica de la nube es sumar 1 al total de caras o de cruces. Un posible argumento sería "¿Es H? Si no, debe ser T". Ello nos daría el siguiente organigrama:



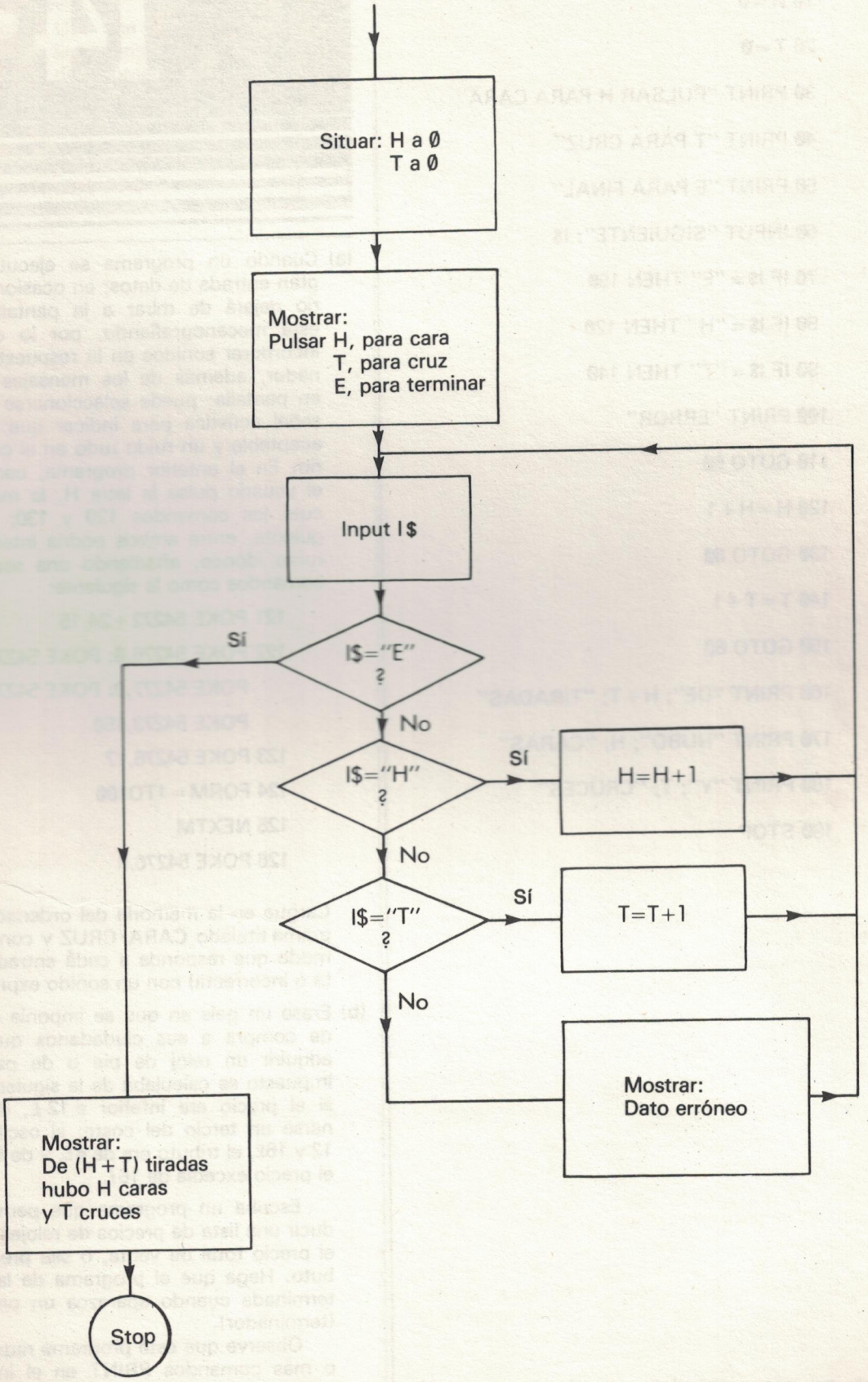

En la práctica, un buen programador no usaría nunca este método, ya que no excluye errores de mecanografía del usuario: si pulsa la J en vez de la H, el programa la contaría como T, a pesar de que evidentemente no es lo que quería el usuario, ya que la T queda apartada.

Es preferible excluir la posibilidad de errores, como en el siguiente ejemplo:



Un programa que permite al usuario cometer errores sin resultados desastrosos, se dice que es robusto.

Finalmente, concretamos la nube de modo que nos dé un informe de tres líneas al final del programa:



He aquí el programa. El código del bucle principal es algo complicado, pero ello es inevitable, porque hemos de forzar un organigrama bidimensional en una serie única de instrucciones.

```
10 H=0
20 T=0
30 PRINT "PULSAR H PARA CARA"
40 PRINT "T PARA CRUZ"
50 PRINT "E PARA FINAL"
60 INPUT "SIGUIENTE": I$
70 IF I$="E" THEN 160
80 IF I$="H" THEN 120
90 IF I$="T" THEN 140
100 PRINT "ERROR"
110 GOTO 60
120 H=H+1
130 GOTO 60
140 T=T+1
150 GOTO 60
160 PRINT "DE"; H+T; "TIRADAS"
170 PRINT "HUBO"; H; "CARAS"
180 PRINT "Y"; T; "CRUCES"
190 STOP
```



# EXPERIMENTO 14·2

(a) Cuando un programa se ejecuta con una gran entrada de datos; en ocasiones el usuario dejará de mirar a la pantalla mientras está mecanografiando, por lo que es útil incorporar sonidos en la respuesta del ordenador, además de los mensajes mostrados en pantalla: puede seleccionarse una alegre señal acústica para indicar que el dato es aceptable y un ruido rudo en el caso contrario. En el anterior programa, cada vez que el usuario pulsa la letra H, la máquina ejecuta los comandos 120 y 130; por consiguiente, entre ambos podría intercalarse un ruido idóneo, añadiendo una secuencia de comandos como la siguiente:

```
121 POKE 54272+24,15
122 POKE 54276,0: POKE 54278,255:
    POKE 54277,0: POKE 54272,90:
    POKE 54273,150
123 POKE 54276,17
124 FORM=1TO100
125 NEXTM
126 POKE 54276,0
```

Cargue en la memoria del ordenador el programa titulado CARA/CRUZ y complételo de modo que responda a cada entrada (correcta o incorrecta) con un sonido expresivo.

(b) Erase un país en que se imponía un tributo de compra a sus ciudadanos que querían adquirir un reloj de pie o de pared; este impuesto se calculaba de la siguiente forma: si el precio era inferior a 12 £, debía abonarse un tercio del coste; si oscilaba entre 12 y 16£, el tributo era de 4£, y de 6£ cuando el precio excedía de 16£.

Escriba un programa que permita introducir una lista de precios de relojes e indique el precio total de venta, o sea precio + tributo. Haga que el programa de la lista por terminada cuando aparezca un precio = 0 (terminador).

Observe que este programa requerirá uno o más comandos PRINT en el interior del bucle y que no se precisa bloque de suma

final. Es indispensable preparar un buen organigrama para elaborar las instrucciones.

(c) Escriba un programa que permita introducir una serie de cantidades, terminando con 0, y muestre la mayor.
Sugerencias: Use una variable que registre la cantidad más elevada en cada momento, actualizada en cada ciclo de ejecución del bucle.

| *Fin del Experimento 14.2* | |
|---|---|



Ahora compruebe sus soluciones con las del Apéndice B.

# UNIDAD: 15


| | | |
|---|---|---|
| EXPERIMENTO 15.1 | PAG. | 121 |
| EXPERIMENTO 15.2 | | 126 |
| EXPERIMENTO 15.3 | | 126 |
| EXPERIMENTO 15.4 | | 129 |

En esta unidad de estudio se tratan tres características importantes del lenguaje BASIC de Commodore que son de utilidad en juegos, ejercicios tipo cuestionarios y programas en que la máquina y el usuario trabajan en estrecha colaboración.

Para empezar, ejecutaremos el programa "REACCION". El tiempo de reacción de una persona se mide por su rapidez en reaccionar a un acontecimiento inesperado.

La mayoría de la gente reaccionan en un tiempo de Ø,2 a Ø,3 segundos, a condición de que presten atención. Cuando el tiempo es menor de Ø,2 segundos, se trata de personas muy rápidas, mientras que si excede de Ø,3 segundos refleja cierta lentitud, con respecto al promedio.



# EXPERIMENTO 15·1

Cargue el programa REACCION y ejecútelo para medir su tiempo de reacción. Opere el programa varias veces y prescinda de los resultados obtenidos las dos o tres primeras veces, ya que no serán representativos. Repita la ejecución del programa hasta que lo comprenda a la perfección y pueda usarlo sin problema para medir el tiempo de reacción de otra persona que no posea conocimiento de ordenadores.

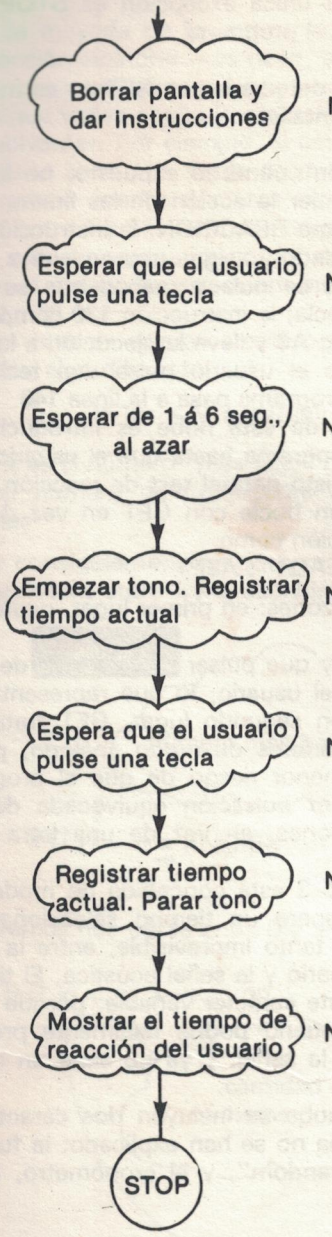
Quizás observará tres aspectos del programa que no son obvios a primera vista:

Primero, cuando las instrucciones dicen cualquier tecla, su significado es literal y puede pulsarse no importa qué tecla, aunque sea de funciones, como RETURN o INST DEL.

Segundo, el tiempo que tiene que esperar antes de oir el tono varía cada vez: oscila entre 1 y 6 seg., de modo que no puede predecirse.

Tercero, si pulsa una tecla antes de que suene el tono, en la pantalla aparece el mensaje "Demasiado Pronto".

Examine ahora el programa detenidamente para ver su funcionamiento. A continuación se reproduce su organigrama y un listado del programa:




Nube 1

```
10 REM PROGRAMA TIEMPO DE REACCION
20 PRINT " SHIFT " CLR HOME
30 PRINT "PARA MEDIR SU"
40 PRINT "TIEMPO DE REACCION"
50 PRINT "PULSE UNA TECLA Y"
60 PRINT "ESPERE A OIR UN TONO"
70 PRINT "CUANDO LO OIGA"
80 PRINT "PULSE UNA TECLA"
90 PRINT "LO MAS RAPIDO"
95 PRINT "QUE PUEDA"
100 PRINT "BUENA SUERTE!"
```

Nube 2

```
110 REM ESPERA UNA TECLA
120 GET A$
130 IF A$="" THEN 120
```

Nube 3 (y observe el anterior)

```
140 REM ESPERA UN TIEMPO ALEATORIO
143 PRINT
145 PRINT "ESPERE EL TONO!"
148 PRINT
150 Q=TI+INT (60+301*RND(0))
160 GET A$
170 IFA$<>"THEN 340
180 IF TI<Q THEN 160
```

Nube 4

```
190 REM COMENZAR TONO Y ANOTAR TIEMPO
200 POKE 54295,0
202 POKE 54296,15
20 POKE 54278,240
207 POKE 54277,16
210 POKE 54273,100
212 POKE 54276,33
220 X=TI
```

Nube 5

```
230 REM ESPERA UNA TECLA
240 GET A$
250 IFA$="THEN 240
```

Nube 6

```
260 REM CALCULAR RESULTADO Y PARAR TONO
270 R=TI
280 POKE 54276,0
290 POKE 54296,0
```

Nube 7

```
300 REM EXHIBIR RESULTADO
310 PRINT "SU TIEMPO DE REACCION"
320 PRINT "ES: "INT (((R-X/60*1000)+5/1000: "SEGUNDOS"
```

Parte de la nube 3

```
330 STOP
340 PRINT "DEMASIADO PRONTO!"
350 STOP
```



Dream commodore

El anterior listado del programa REACCION se ha dividido en secciones, a fin de ilustrar la construcción de las sucesivas nubes del organigrama.

La primera nube (líneas nº 10 a 100) está integrada exclusivamente por comandos PRINT, por lo que no requiere explicación.

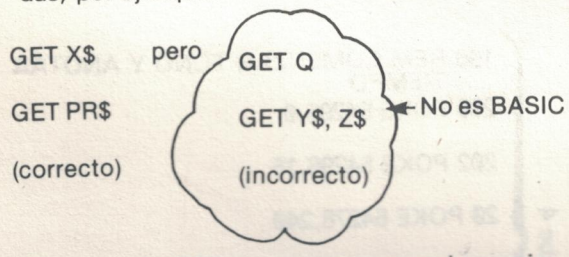
La segunda nube, línea nº 110 a 130, determina una pausa en la ejecución del programa para que el usuario pueda pulsar una tecla; la operación se reanuda tan pronto como se verifica dicha pulsación. Se usa un comando que no se ha tratado todavía:

GET A$

GET es parecido a INPUT; al igual que éste transfiere información desde el teclado al ordenador, pero con importantes diferencias:



1. La palabra clave GET puede ir seguida sólo de 1 nombre de variables de cadena. Los nombres de variables numéricas no son válidas; por ejemplo:




2. El comando GET no espera a que el usuario realice acción alguna; simplemente examina el teclado en el momento en cuestión e indica que tecla se ha pulsado desde la ejecución del último comando GET o INPUT: si se ha pulsado una tecla, adopta la forma de una cadena de 1 carácter y se almacena en la variable mencionada en la instrucción GET; si no se ha pulsado ninguna nueva tecla, la variable se asigna en una cadena nula, ésta es una cadena sin caracteres, que se escribe normalmente"". Para ilustrar esta regla, imaginemos que inicializamos el ordenador con el siguiente programa en bucle y tratemos de ver lo que ocurre dentro de la máquina:

```
10 GET X$
20 GOTO 10
```

El ordenador ejecutará este bucle unas 50 veces por segundo. En tanto el usuario no pulse una tecla, la variable X$ permanecerá como cadena nula:"".
Supongamos ahora que el usuario pulsa una tecla, por ejemplo la letra U. En la próxima ejecución del comando GET (es decir, antes de que transcurra 1/50 seg.), la variable X$ será ahora la cadena "U". Ello tiene carácter provisional, ya que en el siguiente ciclo del bucle X$ se restituirá a "", hasta que se pulse otra tecla cualquiera. La única excepción a esta regla son las teclas de repetición, como por ejemplo la barra espaciadora.

3. El comando GET no reacciona a las teclas de caracteres de control, como INST DEL RETURN o de movimiento del cursor, como casos especiales; la única excepción es STOP, que interrumpe el programa.

4. El carácter detectado por GET no es mostrado en la pantalla.

Teniendo en cuenta lo expuesto, no le será difícil comprender la acción de las líneas 120 y 130 del programa REACCION: la instrucción 120 examina el teclado y asigna una cadena a la variable A$, que es nula a menos que se haya pulsado una tecla; la instrucción 130 comprueba el contenido de A$ y lleva la ejecución a la línea 120 hasta que el usuario pulsa una tecla, en cuyo caso el programa pasa a la línea 140.

El objetivo de esta nube es introducir una pausa en el programa hasta que el usuario indique que está listo para el test de reacción. ¿Por qué usamos un bucle con GET en vez de una simple instrucción como

INPUT "READY": A$?

Hay dos razones: en primer lugar, cuando se usa INPUT hay que pulsar RETURN después del mensaje del usuario, lo que representa dos pulsaciones; en segundo lugar, GET trata casi todos los caracteres de modo análogo, por lo que hay un menor riesgo de que el programa se estropee por pulsación equivocada de una tecla de funciones, en vez de una letra o un número.

La nube nº 3 está concebida de modo que la máquina espere un tiempo seleccionado al azar, y por lo tanto imprevisible, entre la señal "listo" del usuario y la señal acústica. El tiempo ha de tener este carácter variable, porque de lo contrario el usuario podría fácilmente predecir el instante de la señal, y ya no sería un hecho inesperado.

En dicha nube se incluyen dos características que todavía no se han explicado: la función aleatoria, o "random", y el cronómetro, o "timer" interno.

La función aleatoria se utiliza para que la máquina produzca un número impredecible*. Cada vez que el ordenador elabora la expresión RND (Ø) obtiene un valor aleatorio diferente, entre Ø y 1.

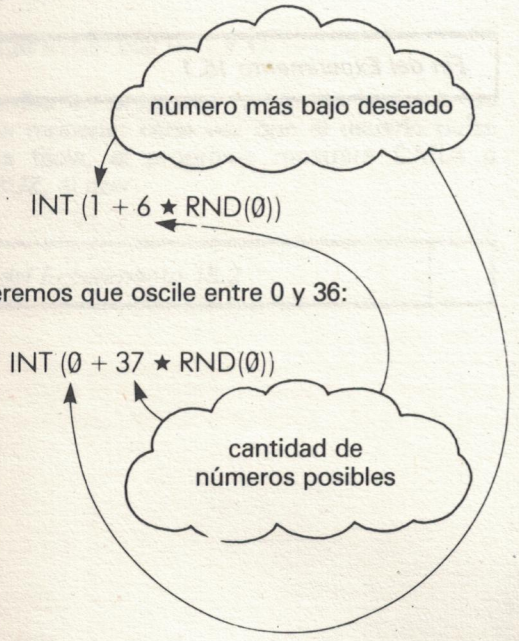
En la mayoría de los casos no necesitamos una fracción aleatoria —es decir, al azar— entre Ø y 1, sino un número entero aleatorio entre límites que varían según la naturaleza del problema a solventar. Por ejemplo, si usted quiere que la máquina imite una tirada de datos hexagonales, requerirá que los números escogidos al azar oscilen entre 1 y 6**, ambos comprendidos; o si quiere usar su ordenador como una ruleta, querrá que el ordenador seleccione números al azar entre Ø y 36.

Para obtener números enteros, usamos una expresión algo diferente:

INT(x + y ★RND(Ø))

donde x es el número "más bajo" deseado.
y es el número de "diferentes posibilidades" deseadas

Por consiguiente, para obtener un número al azar que oscile entre 1 y 6, escribimos:




Si queremos que oscile entre 0 y 36:




* *El número no es realmente impredecible, porque todo lo que ocurre en un ordenador depende de lo que aconteció anteriormente; sin embargo, cada nuevo número "al azar" es derivado del anterior por un complicado proceso de elevarlo a su cuadrado y suprimiendo dígitos del resultado, por lo que a menos que se sepa exactamente su proceso matemático es imposible predecirlo.*

** *Es decir, arrojando un solo dado.*

*Este tipo de expresiones puede incluirse en un bucle, a fin de obtener su elaboración repetida. Escriba en el teclado el siguiente programa, en el que se imitan 12Ø tiradas de un dado:*

```
NEW
1Ø FOR J = 1 TO 12Ø
2Ø S = INT(1 + 6★RND(Ø)
3Ø PRINT S;
4Ø NEXT J
5Ø STOP
```

Ejecute este programa y cuente el número de veces que aparece un 1, 2...6. Luego escriba los resultados en la primera línea de casillas vacía:



| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Nº tiradas (1) | | | | | | |
| Nº tiradas (2) | | | | | | |

¿Es el programa verdaderamente imparcial?

Ahora vuelva a ejecutar el programa y rellene la segunda línea de casillas. Vea que los resultados globales varían, como sería de esperar con un dado de verdad.

La otra importante característica de la nube nº 3 es su cronómetro incorporado, TI. Ya hemos hablado del reloj TI$, que mide el tiempo en horas, minutos y segundos; pero la variable especial TI (que no es una cadena, sino un número) se destina a medir períodos de tiempo mucho más cortos. TI se ajusta automáticamente a cero cuando se conecta el C-64 y, a partir de dicho momento, añade 1 a su contaje cada 1/60 de segundo. Usando TI en una expresión, podemos obtener el valor actual del cronómetro interno, pero no podemos modificarlo como hacíamos con TI$.

Entre el comando:

PRINT TI

La máquina mostrará una cantidad bastante larga (6Ø★6Ø ó 36ØØ por cada minuto que estuvo conectada). Repita el comando y verá como la cifra se ha incrementado en unos cuantos cientos. Dicha cifra no puede editarse ni restituirse a cero.

TI se usa para medir períodos de tiempo en dos formas diferentes, si bien relacionadas entre sí. En ninguna de ellas utilizamos como valor de cálculo el tiempo transcurrido desde el instante que conectamos el ordenador, sino que utilizamos el hecho de que la duración de cualquier período de tiempo es igual a la diferencia entre TI al final de dicho período y el valor que tenía al principio. Por ejemplo, al final de un período de 5 seg., TI será 5★6Ø, o sea 3ØØ, más elevado que al principio. Ello es independiente del tiempo que la máquina haya estado conectada.

En la primera forma de usar el cronómetro interno, la máquina mide un período de tiempo establecido (que ya es conocido) y nos dice cuando ha transcurrido. Este método de temporización es simple: al principio del período, el programa examina a TI y predice lo que será su valor al final del período; luego espera en un bucle hasta que se alcanza (o rebasa) dicho valor. Es muy parecido a lo que usted hace en la cocina cuando quiere hervir un huevo durante 5 minutos: "son las 9,2Ø, así pues lo sacaré a las 9,25".

Para ilustrar este proceso de temporización examine el siguiente programa, que usted puede usar en la cocina, el laboratorio, etc.:



```
1Ø INPUT"CUANTOS MINUTOS";M
2Ø R=TI+M*36ØØ
3Ø IF TI<R THEN 3Ø
4Ø PRINT"YA ES HORA!"
5Ø STOP
```

Ejecute este programa, para practicar, pero especifique un número bajo de minutos, o de lo contrario pasará el tiempo esperando. Cuando estudie el programa verá que TI se eleva continuamente, por lo que al cabo de M*36ØØ, la condición TI<R será falsa.

En la segunda modalidad de uso, pedimos al ordenador que nos diga el tiempo que transcurre desde un momento dado hasta que se produce un acontecimiento dado: hacemos que la máquina registre al principio del período a cronometar, y, cuando el acontecimiento ocurre, la diferencia entre el valor de TI ahora y el valor antes registrado es una medida del tiempo transcurrido, en 1/6Ø de segundo. Es como un alpinista que dijera: "Recuerdo que empecé a subir a las 5; he llegado a la cima a las 11, o sea que he tardado seis horas".

Un programa para medir el tiempo de esta forma contendría comandos como los siguientes:

| | |
|---|---|
| R=TI | (almacena el valor de TI al principio del período) |
| | y luego |
| E=TI | Obtiene el valor de TI al final del período |
| D=E−R | Obtiene la diferencia de tiempo (en 1/6Ø de seg.) |
| S=D/6Ø | Obtiene la diferencia de tiempo (en segundos) |

PRINT"TARDO";S;"SEGUNDOS"

Ahora podemos agrupar esos comandos en la nube nº 3.

Queremos un período de espera entre 1 y 5 seg., es decir entre 6Ø y 3ØØ 1/6Ø seg., que será decidido por máquina de modo impredecible.

INT(6Ø+3Ø1*RND(Ø))

El período de espera es seleccionado justamente antes del comienzo del período, por lo que es conocido de antemano (pero no por el usuario). Utilizamos el primer método de cronometraje, que exige predecir el valor de TI al final del período. El comando 15Ø realiza esta predicción y registra el valor en Q.

Si no necesitáramos nada más, la nube completa sería:

```
15Ø Q=TI+INT(6Ø+3Ø1*RND(Ø))
16Ø IF TI<Q THEN 16Ø
```

En nuestro caso, hemos de comprobar además que el usuario no pulse una tecla antes de que suene el tono. Para comprobar esta posibilidad usamos las instrucciones nº 16Ø, 17Ø, 34Ø y 35Ø.

El resto del programa no plantea dificultades. El valor de TI al principio del período de tiempo de reacción se almacena en X, y las instrucciones 24Ø y 25Ø se utilizan para esperar que el usuario pulse una tecla.

Estudie el programa detenidamente, para comprender todos los comandos.

*Fin del Experimento 15.1*

# EXPERIMENTO 15·2

1. Escriba un programa de cronometraje: cuando el usuario pulsa la tecla B, el programa empieza a medir el tiempo, y cuando pulsa S, se para y muestra el tiempo transcurrido en segundos. El programa debe mostrar en la pantalla mensajes fáciles de interpretar, para que cualquiera pueda ejecutarlos sin explicación previa.

   Sugerencia: use GET y TI.

2. Escriba un programa que imite la tirada de una moneda: cada vez que el usuario pulsa una tecla, el programa mostrará CARA o CRUZ, al azar.

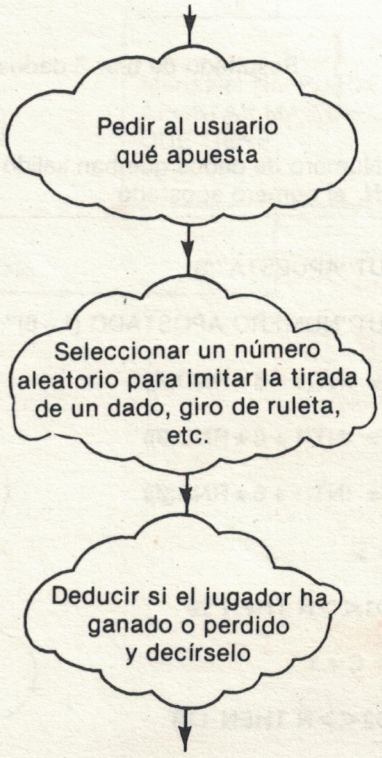
*Fin del Experimento 15.2*

Compare su programa con el ilustrado en el Apéndice B.

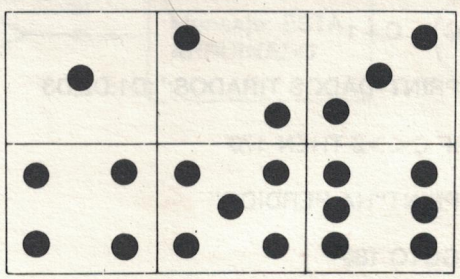
# EXPERIMENTO 15·3



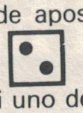
Los números aleatorios son útiles en la programación de juegos al azar, tales como dados, etc., Todos estos programas siguen una pauta común, a saber:




Ilustremos todo eso con un antiguo juego de dados "corona y ancla"*. Se juega con tres dados y una tabla dividida en seis cuadrados:



* Los dados "corona y ancla" suelen tener símbolos diferentes, pero ello no afecta el principio del juego.

El jugador pone su apuesta en un cuadrado cualquiera; por ejemplo, puede apostar 5 Ptas. en el cuadrado de 2 puntos  . Entonces la banca tira los tres dados: si uno de ellos tiene el valor 2 , el jugador retira el doble del dinero que apostó; si son dos los dados con el valor 2 , obtiene el triple, y el cuádruplo, si son los tres con dicho valor. Por otra parte, si ningún dado da 2 , pierde su apuesta.

A continuación se lista el programa para jugar una tirada de dados "corona y ancla". Estudie el glosario de variables, para poder comprenderlo sin problema:



S: Apuesta del jugador
N: Número apostado
D1
D2 } Resultado de tirar 3 dados
D3
C: Número de dados que han salido con N, el número apostado

```
10 INPUT"APUESTA";S
20 INPUT"NUMERO APOSTADO (1-6)";N
30 D1 = INT(1+6*RND(0))
40 D2 = INT(1+6*RND(0))
50 D3 = INT(1+6*RND(0))
60 C = 0
70 IF D1<>N THEN 90
80 C = C+1
90 IF D2<>N THEN 110
100 C = C+1
110 IF D3<>N THEN 130
120 C = C+1
130 PRINT"DADOS TIRADOS:";D1;D2;D3
140 IF C<>0 THEN 170
150 PRINT"HA PERDIDO"
160 GOTO 180
170 PRINT"VD GANA";S*(C+1);"PESETAS"
180 STOP
```

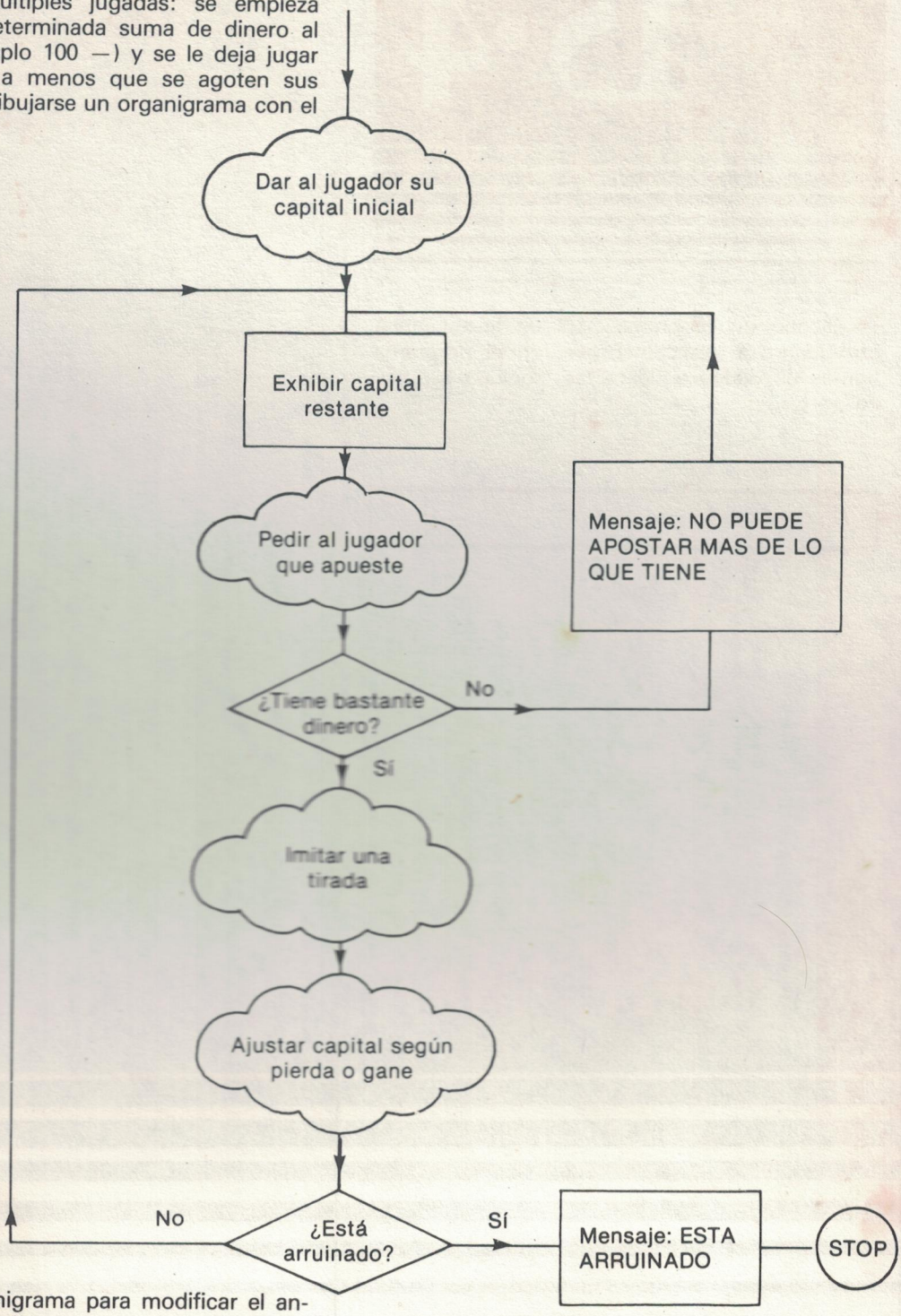
Tira los dados

Cuenta el número de dados con el valor apostado

Muestra los resultados

Muy pocos jugadores se retiran al cabo de la primera jugada. Normalmente tratan de recuperar lo perdido o incrementar sus ganancias hasta que no les queda ni cinco o —muy raramente— salta la banca.

Por consiguiente, conviene que los programas permitan múltiples jugadas: se empieza asignando una determinada suma de dinero al jugador (por ejemplo 100 —) y se le deja jugar en tanto quiere, a menos que se agoten sus recursos. Puede dibujarse un organigrama con el el siguiente:




Usar este organigrama para modificar el anterior programa de "corona y ancla", de modo que el usuario pueda empezar con 100 — y jugar todo el tiempo que quiera. Después de completar el programa, ejecútelo varias veces, para que a la vista de los resultados usted pueda escoger entre ser un simple jugador o tener la banca.

*Fin del Experimento 15.3*

# EXPERIMENTO 15·4



Escriba un programa para imitar cualquier otro juego de azar. Incorpore en el programa figuras de dados o de cartas, sonidos expresivos, etc.

*Fin del Experimento 15.4*

En el Apéndice B se muestra un programa para jugar "craps" (juego al azar americano)

# EPILOGO

# EPILOGO



Le felicitamos por haber llegado al final del curso. Usted ha adquirido amplios conocimientos de los principios de programación, que le permitirán diseñar y escribir programas para una extensa gama de aplicaciones. Esperamos que haya desarrollado también el hábito de planificar previamentes sus programas con eficaces organigramas, glosarios de las variables y notas. Es esta planificación y trabajo metódico lo que hace que un programador competente destaque del resto.

Usted ha recorrido la mitad del camino en su estudio del BASIC. Hay aún muchos problemas que no puede resolver sin incrementar su vocabulario y conocimiento de la sintaxis del lenguaje BASIC. Por ejemplo, quizás quiere programar dibujos animados en la pantalla o clasificar los nombres de personas por orden alfabético o almacenarlos en cinta cassette. Estas tareas y muchas más se explican sistemáticamente en el segundo libro de esta serie, titulado

INTRODUCCION AL BASIC (Parte II)

Este libro está concebido en el mismo estilo que el que acaba de terminar y completará sus conocimientos del lenguaje BASIC.

Programar —como dijimos en un principio— es un tema muy amplio. Ahora que ha recorrido un importante trecho, puede ampliar sus conocimientos de tres formas principales:

(a) Lea tanto como le permita su tiempo sobre ordenadores. La mayoría de revistas sobre informática, sobre todo la publicación "Club COMMODORE", son extremadamente útiles. Los libros editados sobre programación son también valiosos, aún cuando no se refieren específicamente a ordenadores C-64.

(b) Hágase miembro de algún club de informática, que existe en su país.

(c) Dedique tiempo a la programación. Practique constantemente y trate de producir programas perfectos, que sean robustos y puedan ser utilizados por cualquier persona, sin instrucción especial.

Usted ha encontrado una afición fascinante que quizás querrá convertir en su profesión. Recuerde que junto a las ventajas de ser experto en informática hay también una responsabilidad: el esforzarse en que se dé un uso humano y prudente a los ordenadores; nadie quiere una sociedad controlada por ordenadores, con altas tasas de desempleo y escasa libertad. Depende de usted —junto a otros en el mismo ámbito de acción— el evitar la servidumbre del hombre a la máquina.

# APENDICE


| | | |
|---|---|---|
| APENDICE A | PAG. | 133 |
| APENDICE B | | 139 |
| APENDICE C | | 154 |
| APENDICE D | | 155 |

# APENDICE A



El C-64 es un ordenador capaz de realizar cálculos matemáticos de gran escala. En este Apéndice se describen brevemente algunas de las posibilidades matemáticas del C-64. Usted puede usar el C-64 para cálculos matemáticos, científicos o de ingeniería. Algunas de las características descritas a continuación son muy simples y pueden ser comprendidas por cualquiera que recuerde la aritmética elemental que aprendió en la escuela. Otras facetas exigen un conocimiento más profundo, como matemáticas a nivel preuniversitario. Usted no necesita ir más lejos de lo que le permita su dominio de las matemáticas, aunque es preciso que lea todas las secciones de este Apéndice, como hizo antes con todas las unidades de estudio del curso.

## 1. Expresiones

Las expresiones, citadas ya en la Unidad 4, pueden ser de complejidad muy variable. Observe las expresiones contenidas en los siguientes comandos:

A = 34

B = B + 1

C = ((X + Y) — 34.7/(Q—3))★(Z—3)↑2

Los elementos subrayados constituyen expresiones que el C-64 puede calcular para usted.

Las expresiones se componen de tres tipos de elementos:

Valores, que pueden ser variables numéricas o números, como

B,X,Y,34,34.7

Operadores: los signos + — ★/ y ↑

(↑ significa "elevado a la potencia")

Paréntesis (y)

Las expresiones en BASIC se escriben como en álgebra y tienen el mismo significado. Hay cuatro diferencias:

- La expresión en BASIC es con mayúsculas en vez de minúsculas.
- La exponenciación ("elevación a la potencia") debe indicarse con el signo ↑, ya que la pantalla C-64 no permite escribir pequeños números a media línea; así en vez de "$3^2$", debe ponerse "3↑2"
- La multiplicación debe indicarse con el signo ★. En BASIC se escribe "3★A", en vez de "3A", como en álgebra. El no observar esta regla puede dar lugar a errores difíciles de localizar: si usted escribe BA cuando quiere decir B★A, la máquina asumirá que se trata de una nueva variable, denominada BA. No mostrará ningún mensaje de error, pero la respuesta será errónea.
- La división se indica en la forma A/B en vez de $\frac{A}{B}$ Si el numerador o el denominador de la fracción es una expresión complicada, debe delimitarse con paréntesis. La forma correcta de escribir $\frac{3+5}{7+8}$ en BASIC es (3+5)/(7+8). Si usted omite los paréntesis y escribe 3+5/7+8, la máquina tratará la expresión como $3 + \frac{5}{7} + 8$.

Cuando C-64 calcula una expresión, calcula primero la exponenciación, luego las multiplicaciones, divisiones y, finalmente, sumas y restas, de izquierda a derecha. Los elementos entre paréntesis se calculan primero. Este orden de cálculo se denomina "regla de precedencia" y equivale a los procesos seguidos en álgebra corriente.

El valor de los números en las expresiones no han de ser necesariamente enteros, sino también decimales. El C-64 calcula con una precisión de unos 8 decimales, lo que significa que fracciones tales como 1/3 o 1/7 pueden representarse con exactitud. En ocasiones puede haber pequeños errores de redondeado, haciendo que algunos

comandos aritméticos no den exactamente el resultado que usted esperaba, como "6,99999998" en vez de 7.

Para comprobar su comprensión de las expresiones BASIC, calcule los siguientes ejemplos, suponiendo que el valor de X es 3 y el de P es 7. Anote el resultado calculando en la columna "Resultado Previsto".



| *COMANDO* | *RESULTADO PREVISTO* | *RESULTADO REAL* |
|---|---|---|
| PRINT 3 + 12 — 6 — 4 | | |
| PRINT 4 + 3 ★ 2 | | |
| PRINT X + P — 3 | | |
| PRINT 5 + 12 / 6 — 3 | | |
| PRINT 11 / 5 — 7 / 4 | | |
| PRINT 4 ↑ 2 — 2 ↑ 4 | | |
| PRINT 3 + 2 ↑ 3 — 3 ↑ 2 | | |
| PRINT 2 ↑ X — P | | |
| PRINT 3 + 12 — (6 — 4) | | |
| PRINT 5 + 12 / (6 — 3) | | |
| PRINT (P + X) ↑ (1 — X) | | |
| PRINT 4 ↑ 2 — 3 ↑ Ø | | |
| PRINT (P ↑ 2 — X ↑ 2) / 3 | | |

Ahora compruebe los resultados calculados con los que va a dar el C-64. Acuérdese de asignar los valores citados a P y X antes de empezar.

En BASIC, las expresiones se usan principalmente con los comandos PRINT y LET. A continuación se lista un simple programa que introduce dos cifras U y V con INPUT y da un valor F calculando con arreglo a la siguiente fórmula:

$$\frac{1}{F} = \frac{1}{V} + \frac{1}{U} \text{ , ó } F = \frac{1}{\frac{1}{V} + \frac{1}{U}}$$

```
10 INPUT "V"; V
20 INPUT "U"; U
30 PRINT "F="; 1/(1/V+1/U)
40 STOP
```

## Ejemplo 1

Escriba un programa que permita introducir dos valores V y R y que muestre el valor de la fórmula $A = \frac{V^2}{R}$



## Ejemplo 2

Escriba un programa que de los valores de la fórmula $y = \frac{1}{1 + X^2}$, con valores de x entre 0 y 2, en incrementos de 0,2

(Ayuda: use un bucle FOR como el siguiente:

```
FOR X = 0 TO 2 STEP 0.2
.....
NEXT X      )
```

(Las soluciones se dan al final del Apéndice B)

## 2. Funciones Típicas

Al igual que muchas calculadoras, el C-64 incorpora una serie de funciones científicas; una de ellas es la raíz cuadrada; está abreviada a la forma SQR y puede incluirse en expresiones como las siguientes:

```
PRINT SQR(5)
```

o

```
PRINT SQR (B↑2 + C↑2)
```

La cantidad entre paréntesis se llama el "argumento" de la función. En el caso de la función SQR, el argumento debe ser cero o positivo.

He aquí un programa que da la raiz cuadrada de todos los números entre 100 y 115:

```
10 PRINT "N","SQR(N)"
20 FOR N = 100 TO 115
30 PRINT N; SQR(N)
40 NEXT N
50 STOP
```

## Ejemplo 3

Si las longitudes de los tres lados de un triángulo son a, b y c, el área del triangulo puede obtenerse utilizando la fórmula

$$\sqrt{s(s-a)(s-b)(s-c)}$$

donde s es el semiperímetro, o sea $(a+b+c)/2$.

Escriba un programa que permita introducir tres números con instrucciones INPUT. Si pueden ser los lados de un triángulo real, el programa muestra en la pantalla el área del triángulo; de lo contrario (por ej. si los números son 1, 1, 10), el C-64 dará el oportuno mensaje.

(Nota: si las líneas no forman un triángulo, el valor de $s(s-a)(s-b)(s-c)$ es negativa).

A continuación se relacionan algunas de las funciones matemáticas más importantes. Fíjese atentamente en las letras con que se representan, pero no trate de aprendérselas de memoria, ya que siempre puede consultar esta lista cuando precise utilizarlas:

| | |
|---|---|
| SIN(X) | Funciones trigonométricas. Los argumentos deben ir dados en radianes |
| COS(X) | (1 grado = $\pi/180$ radianes) |
| TAN(X) | |
| ATN(X) | La tangente de arco de X. El resultado es en radianes, entre $-\pi/2$ y $\pi/2$. |
| LOG(X) | El logaritmo natural de X (logaritmo en base e). X debe ser positivo. |
| EXP(X) | Equivale a $e^x$ |
| ABS(X) | El módulo de X (VALOR ABSOLUTO DE X). |
| INT(X) | El mayor número entero igual o menor que X. Véase que:<br>INT (3.5) = 3<br>INT (−3.5) = −4 |



Ahora compruebe la solución que usted ha obtenido con la del Apéndice B.

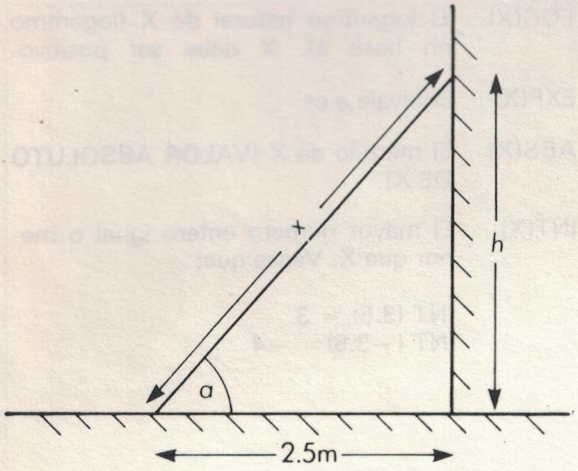
Usted puede usar el símbolo del teclado $\pi$ en vez del número 3.14159265...

A continuación se expone un ejemplo de cómo usar algunas de esas funciones.

Una escalera de mano se alarga desde 4 m. a 5 m., en incrementos de 20 cm. Se coloca contra una pared vertical, a 2,5 m. de su base. Escriba un programa que muestre en la pantalla el ángulo de la escalera con el suelo horizontal en cada una de sus 6 posibles longitudes.

Primero ejecutamos las ecuaciones matemáticas, usando un diagrama. Usamos x para indicar la longitud de la escalera y h para la altura de la parte superior de la escalera, y a para el ángulo con la horizontal.






$h = \sqrt{x^2 - (2.5)^2}$ (según Pitágoras)

$a = \arctan(h/2.5)$ (en radianes)

o $a = (180/\pi) \star \arctan(h/2.5)$ en grados

Entonces escribimos el programa, que posee una estructura de bucle simple:

```
10 PRINT"LONGITUD","ANGULO"
20 FOR X=4 TO 5 STEP 0.2
30 H=SQR(X↑2-2.5↑2)
40 A=(180/π)★ATN(H/2.5)
50 PRINT X,A
60 NEXT X
70 STOP
```

Una de las funciones más útiles es INT. Podemos usarla para averiguar si un número divide a otro con exactitud: Si X es un múltiplo exacto de Y, la condición

$$X/Y = INT(X/Y)$$

será verdadera; en caso contrario, será falsa.

Un número es "primo" cuando puede ser dividido sólo por sí mismo o por 1. El siguiente programa calcula y exhibe los números primos desde el 3 hasta cualquier valor fijado por el usuario:

```
10 INPUT "VALOR MAS ALTO";H
20 FOR N=3 TO H
30 FOR J=2 TO N-1
40 IF N/J=INT (N/J) THEN 70
50 NEXT J
60 PRINT N;
70 NEXT N
80 STOP
```

## Ejemplo 4

Estudie el anterior programa para el cálculo de números primos (de ser necesario efectúe su rastreo) y averigüe como opera. Ejecútelo y adjudique algún valor de cronometraje a H (por ejemplo 500).

Este método de calcular los números primos es en realidad muy lento. Introduzca modificaciones, para que pueda ser ejecutado más rápidamente.

Ayuda: (a) Ningún número par, excepto 2, puede se número primo.

(b) Al investigar posibles coeficientes, es suficiente ir hasta la raiz cuadrada del número.



Ahora compruebe su solución con la ilustrada en el Apéndice B.

# APENDICE B



## CLAVE PARA INTERPRETAR LOS LISTADOS DEL CURSO DE INTRODUCCION AL BASIC

TODOS los listados que se publican a continuación han sido ejecutados en el COMMODORE 64, para mejorar su legibilidad por parte del usuario se les ha sometido a ciertas modificaciones mediante un programa escrito especialmente para ello. Para los programas destinados a este ordenador que usa frecuentemente las posibilidades gráficas del teclado— se han sustituido los símbolos gráficos que aparecen normalmente en los listados por una serie de letras entre corchetes ([]) que indican la secuencia de teclas que se deben pulsar para obtener el caracter deseado. A continuación se da una tabla para aclarar la interpretación de las indicaciones entre corchetes:

[CRSRD) = Tecla cursor hacia abajo (sin SHIFT)
[CRSRU] = Tecla cursor hacia arriba (con SHIFT)
[CRSRR] = Tecla cursor a la derecha (sin SHIFT)
[CRSRL] = Tecla cursor a la izquierda (con SHIFT)
[HOME] = Tecla CLR/HOME (sin SHIFT)
[ CLR ] = Tecla CLR/HOME (con SHIFT)
[ SPC ] = Barra espaciadora

Las indicaciones [ BLK ] a [ YEL ] corresponden a la pulsación de las teclas de 1 a 8 junto a la tecla CTRL, lo mismo sucede con [RVSON] y [RVSOF] respecto a la tecla CTRL y las teclas 9 y Ø.

El resto de las indicaciones constan de la parte COMM o SHIF seguidas de una letra, número o símbolo —por ejemplo [COMM+] o [SHIFA]— esto indica que para obtener el gráfico necesario en el programa deben pulsarse simultáneamente las teclas COMMODORE (la que lleva el logotipo) o una de SHIFT y la tecla indicada por la letra, el número o el símbolo, en el ejemplo anterior: COMMODORE y + o SHIFT y A, respectivamente.
Además debe notarse que dentro de los mismos corchetes se incluye un número que indica cuantas veces debe pulsarse la tecla correspondiente, así [16CRSRD] equivale a 16 pulsaciones de la tecla de cursor abajo.

# UNIDAD: 7

Experimento 7.1:

a) T,T,T,T,F,F,F
b) F,F,T,T

Experimento 7.2:

| *Variable de control* | *Valor inicial* | *Valor final* | *Incremento* | *Número de ciclos* |
|---|---|---|---|---|
| X$ | "A" | "ABBB" | "B" | 4 |
| P | Ø | 1Ø | +1 | 11 |
| Y$ | "Z" | "ZXYXY" | "XY" | 3 |
| R | 5 | 14 | 3 | 4 |
| C | 27 | 7 | —5 | 5 |

Experimento 7.3:

```
(1)  1Ø P$= "★"
     2Ø PRINT P$
     3Ø P$= P$+"★"
     4Ø IF P$<> "★★★★★★★★★★★"
        THEN 2Ø
     5Ø STOP

(2)  1Ø PRINT "LIBRAS","DOLARES"
     2Ø PRINT
     3Ø P= 1Ø
     4Ø PRINT P, 1.77★P
     5Ø P=P+2
     6Ø IF P< 32 THEN 4Ø
     7Ø STOP

(3)  1Ø PRINT "CENT", "FAHR"
     2Ø PRINT
     3Ø C= 15
     4Ø F= 1.8★C+32
     5Ø PRINT C, F
     6Ø C=C+1
     7Ø IF C < 31 THEN 4Ø
     8Ø STOP
```

# UNIDAD: 8

Experimento 8.1:

a) CONTADOR DE PROGRAMA ~~10 20 30 40 50~~ 60

VARIABLES X: 5 Y: 7 Z: 12 W: 2

```
5 7 12 2
BREAK IN 60
READY
```

```
10 X=5
20 Y=7
30 Z=X+Y
40 W=Y—X
50 PRINT X;Y; Z;W
60 STOP
```

CONTRADOR DE PROGRAMA ~~10 20 30 40 50 30 40 50~~ 60

VARIABLES Q: ~~1 2~~ 3

```
ME QUIERE
NO ME QUIERE
NO ME QUIERE
BREAK IN 60
READY
```

```
10 Q = 1
20 PRINT "ME QUIERE"
30 PRINT "NO ME QUIERE"
40 Q = Q + 1
50 IF Q<3 THEN 30
60 STOP
```

Experimento 8.2:

a) La línea 50 debería ser: 50 IF G<11 THEN 30

b) La línea 30 debería ser: 30 A$ = A$ = "★"

Experimento 8.3:

c) Línea 20: PRINT (no PRINT)
Después de la línea 40 no debe haber RETURN
Línea 60: IF X<13 THEN 40 (NOT X>13)
Línea 70: STOP (no STOP)



Dream commodore

# UNIDAD: 9

Experimento 9.2:

```
PROGRAMA: EXPERIMENTO 9.2


10 POKE53280,4
15 POKE53281,7
20 PRINT"[1CLR][1BLK]"
30 PRINT"[1CLR][7CRSRR][9CRSRD]"
;TI$
40 FORI=1TO600:NEXT:GOTO 30
```

Experimento 9.3:

```
PROGRAMA: EXPERIMENTO 9.3


10 POKE53280,14
15 POKE53281,6
20 PRINT"[1CLR]";
30 J=1
40 PRINT"[1RVSON][12CRSRR][1WHT]
[1SPC][1RED][4SPC][1WHT][1SPC]"
50 J=J+1
60 IFJ<11THEN 40
70 PRINT"[1RVSON][1WHT][13SPC][1
RED][4SPC][1WHT][23SPC]";
80 J=1
90 PRINT"[1RVSON][1RED][40SPC]";

100 J=J+1
110 IFJ<4THEN90
120 PRINT"[1RVSON][1WHT][13SPC][
1RED][4SPC][1WHT][23SPC]";
130 J=1
140 PRINT"[1RVSON][12CRSRR][1WHT
][1SPC][1RED][4SPC][1WHT][1SPC]"

150 J=J+1
160 IFJ<10THEN140
170 PRINT"[1RVSON][12CRSRR][1WHT
][1SPC][1RED][4SPC][1WHT][1SPC]"
;
180 GOTO180
```



# UNIDAD: 10

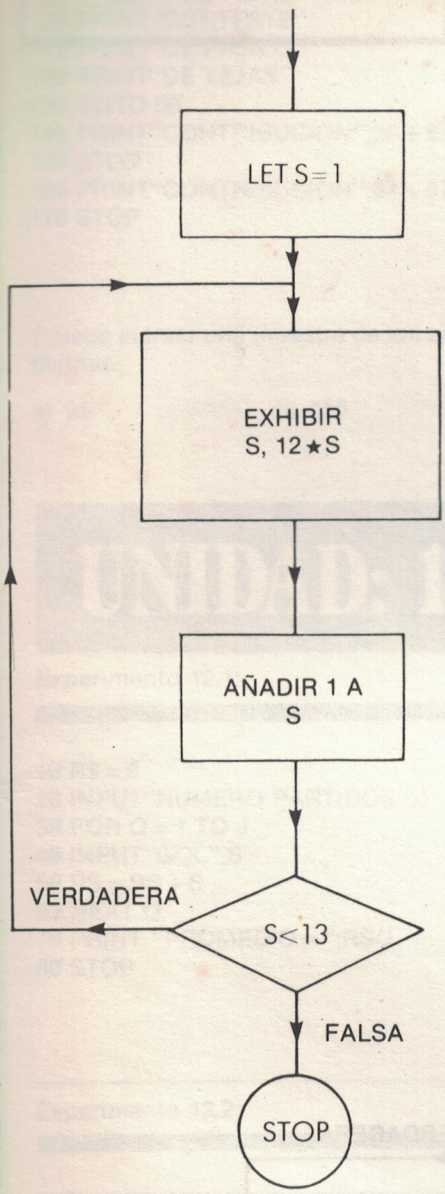
Experimento 10.2:

a)
```
1Ø PRINT"PROGRAMA TABLAS"
2Ø INPUT"POR";N
3Ø X = 1
4Ø PRINT X;"POR";N;"ES";N * X
5Ø X = X + 1
6Ø IF X<13 THEN 4Ø
7Ø STOP
```

(b)
```
1Ø PRINT"ESCRIBA SU"
2Ø INPUT"APELLIDO";S$
3Ø PRINT "ESCRIBA EL"
4Ø PRINT"NOMBRE DE PILA DE SU HIJO"
5Ø INPUT C$
6Ø PRINT"SU HIJO SE LLAMA"
7Ø PRINT C$ + "  " + S$
8Ø STOP
```

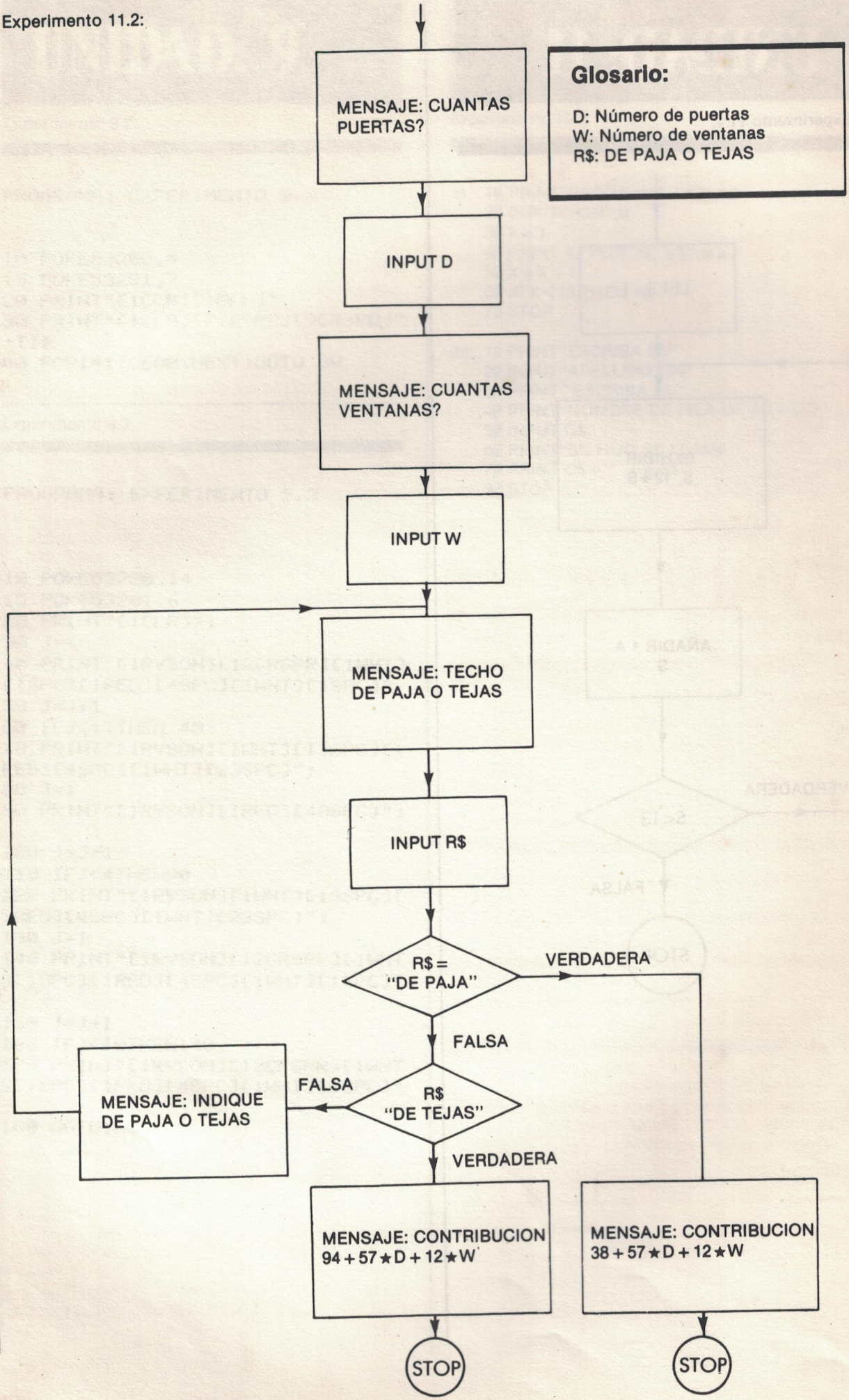
# UNIDAD: 11

Experimento 11.1:



Experimento 11.2:





```
1Ø REM CONTRIBUCIONES EN RURITANIA
2Ø PRINT"PROGRAMA CONTRIBUCION"
3Ø INPUT"CUANTAS PUERTAS";D
4Ø INPUT"CUANTAS VENTANAS";W
5Ø PRINT"ES LA CUBIERTA"
6Ø PRINT"DE PAJA O"
7Ø INPUT"DE TEJAS";R$
8Ø IF R$ = "DE PAJA" THEN 14Ø
9Ø IF R$ = "DE TEJAS" THEN 16Ø
1ØØ PRINT"CONTESTE"
11Ø PRINT"DE PAJA O"
12Ø PRINT"DE TEJAS"
13Ø GOTO 5Ø
14Ø PRINT"CONTRIBUCION";38 + 57★D + 12★W
15Ø STOP
16Ø PRINT"CONTRIBUCION";94 + 57★D + 12★W
17Ø STOP
```

Pruebe extraer una muestra de los siguientes problemas:

a) 95 b) 155 c) 364

# UNIDAD: 12

Experimento 12.1:

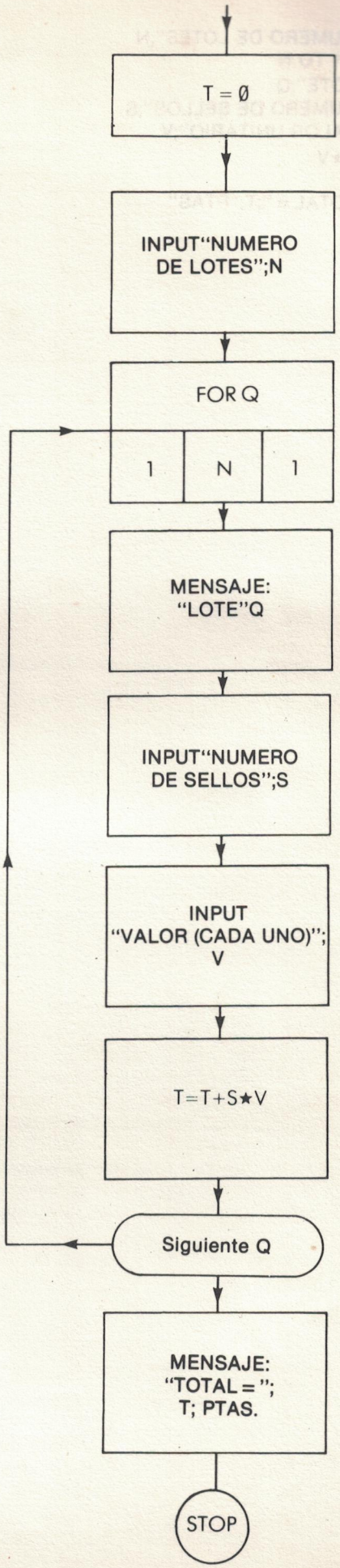
```
1Ø R$ = Ø
2Ø INPUT"NUMERO PARTIDOS";J
3Ø FOR Q = 1 TO J
4Ø INPUT"GOL";S
5Ø RS = RS + S
6Ø NEXT Q
7Ø PRINT "PROMEDIO = ";RS/J
8Ø STOP
```

El orden es indiferente

Experimento 12.2:

## Glosario:

N: Número de lotes
S: Número de sellos por lote
V: Valor de cada sello en cada lote
T: Total acumulativo
Q: Para contar el número de lotes





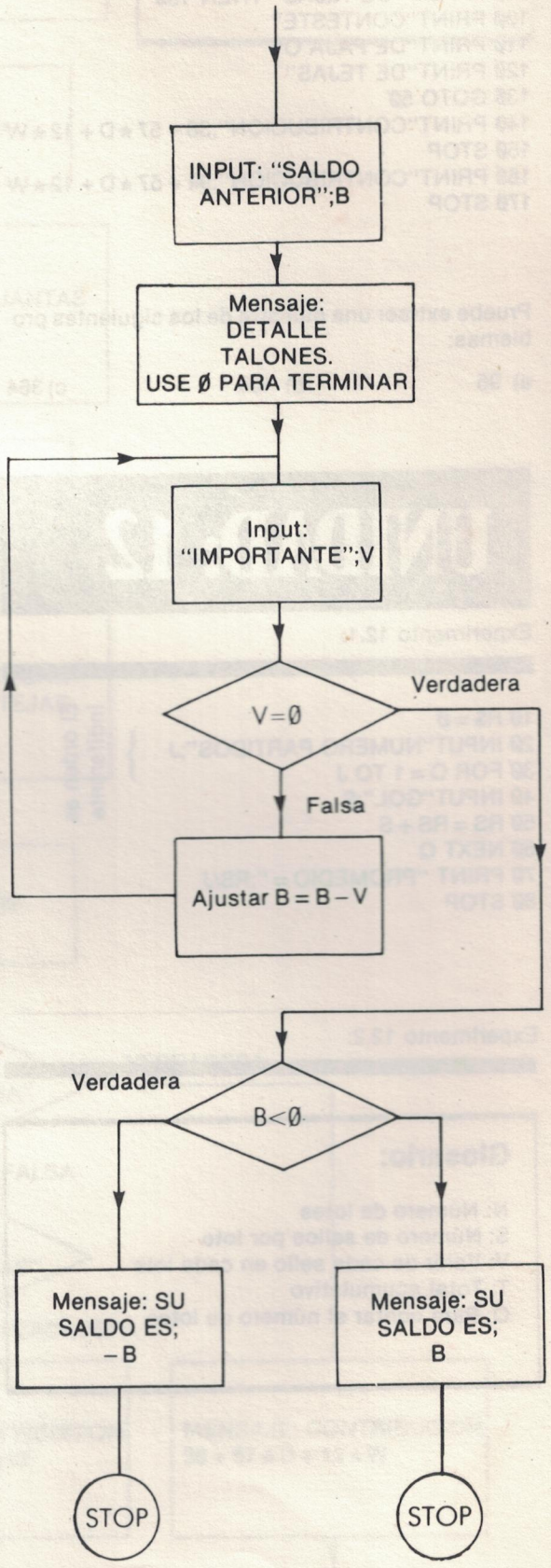

Dream commodore

```
10 T=0
20 INPUT"NUMERO DE LOTES";N
30 FOR Q=1 TO N
40 PRINT"LOTE";Q
50 INPUT"NUMERO DE SELLOS";S
60 INPUT"VALOR UNITARIO";V
70 T=T+S*V
80 NEXT Q
90 PRINT"TOTAL=";T;"PTAS"
100 STOP
```



# UNIDAD: 14

Experimento 14.1:

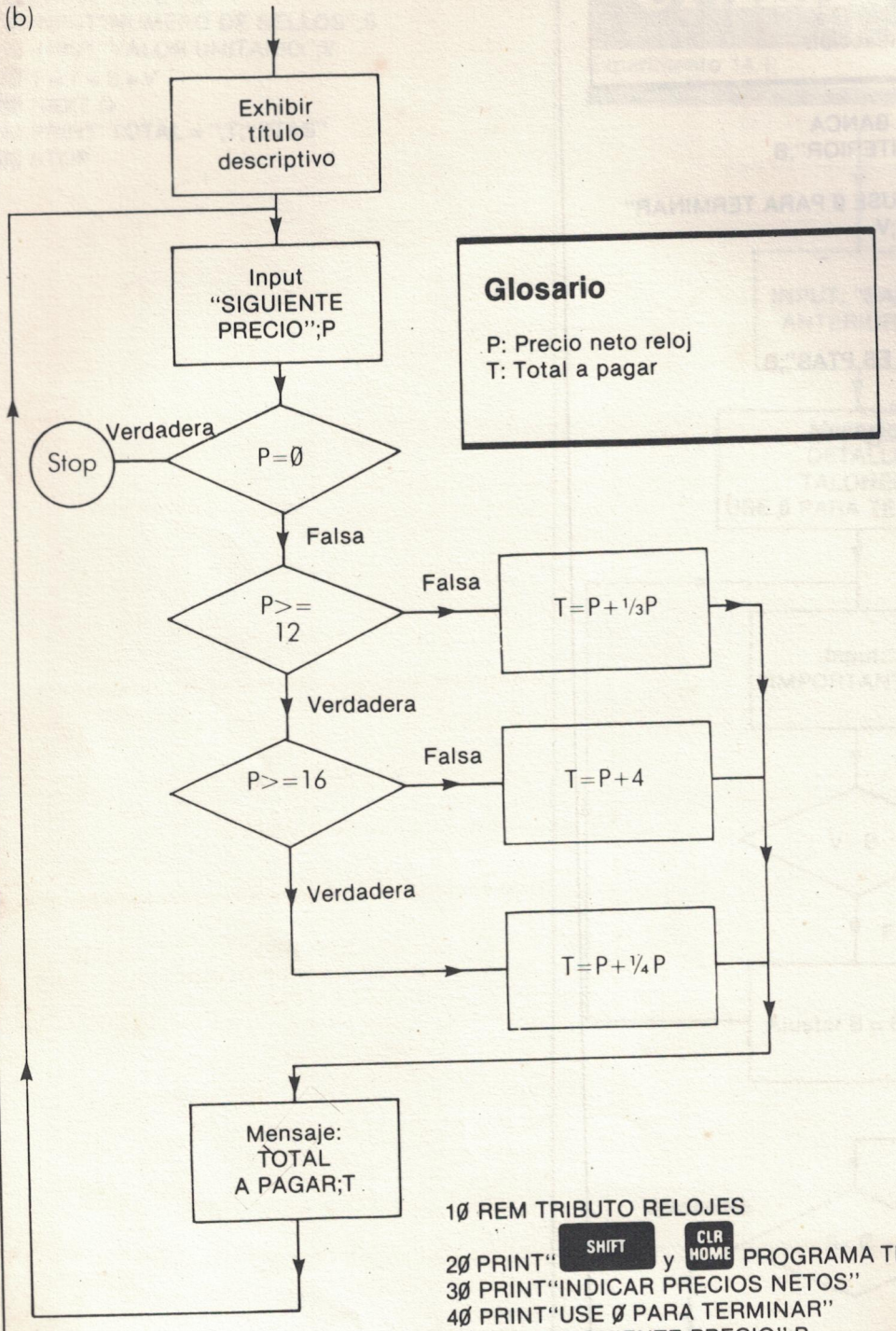


**Glosario:**

B: Saldo actual
V: Cada nueva transacción

```
1Ø REM PROGRAMA BANCA
2Ø INPUT"SALDO ANTERIOR";B
3Ø PRINT"DETALLE"
4Ø PRINT"TALONES.USE Ø PARA TERMINAR"
5Ø INPUT"IMPORTE";V
6Ø IF V=Ø THEN 9Ø
7Ø B=B—V
8Ø GOTO 5Ø
9Ø IF B<Ø THEN 12Ø
1ØØ PRINT"SU SALDO ES PTAS";B
11Ø STOP
12Ø PRINT"SU SALDO"
13Ø PRINT"ES PTAS";—B
14Ø STOP
```

(b)



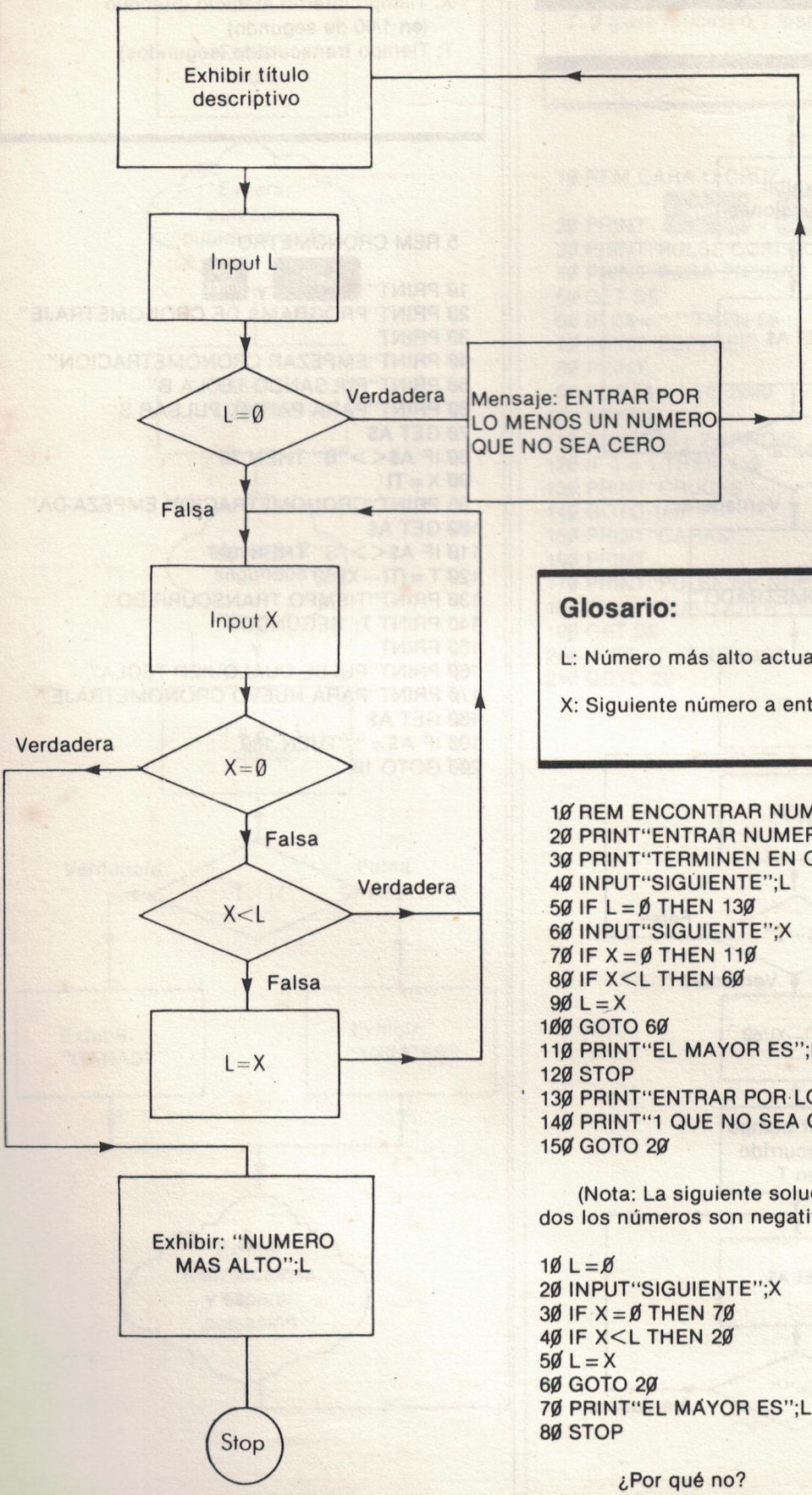

**Glosario**

P: Precio neto reloj
T: Total a pagar



```
1Ø REM TRIBUTO RELOJES
2Ø PRINT" SHIFT y CLR HOME PROGRAMA TRIBUTO RELOJES"
3Ø PRINT"INDICAR PRECIOS NETOS"
4Ø PRINT"USE Ø PARA TERMINAR"
5Ø INPUT"SIGUIENTE PRECIO";P
6Ø IF P=Ø THEN 18Ø
7Ø IF P>=12 THEN 1ØØ
8Ø T=P+ (1/3)*P
9Ø GOTO 14Ø
1ØØ IF P>=16 THEN 13Ø
11Ø T=P+4
12Ø GOTO 14Ø
13Ø T=P+(1/4)*P
14Ø PRINT"TOTAL A PAGAR"
15Ø PRINT"ES";T
16Ø PRINT
17Ø GOTO 5Ø
18Ø STOP
```

c)




**Glosario:**

L: Número más alto actual

X: Siguiente número a entrar

```
1Ø REM ENCONTRAR NUMERO MAS ALTO
2Ø PRINT"ENTRAR NUMERO QUE"
3Ø PRINT"TERMINEN EN CERO"
4Ø INPUT"SIGUIENTE";L
5Ø IF L=Ø THEN 13Ø
6Ø INPUT"SIGUIENTE";X
7Ø IF X=Ø THEN 11Ø
8Ø IF X<L THEN 6Ø
9Ø L=X
1ØØ GOTO 6Ø
11Ø PRINT"EL MAYOR ES";L
12Ø STOP
13Ø PRINT"ENTRAR POR LO MENOS"
14Ø PRINT"1 QUE NO SEA CERO"
15Ø GOTO 2Ø
```

(Nota: La siguiente solución no operará si todos los números son negativos)

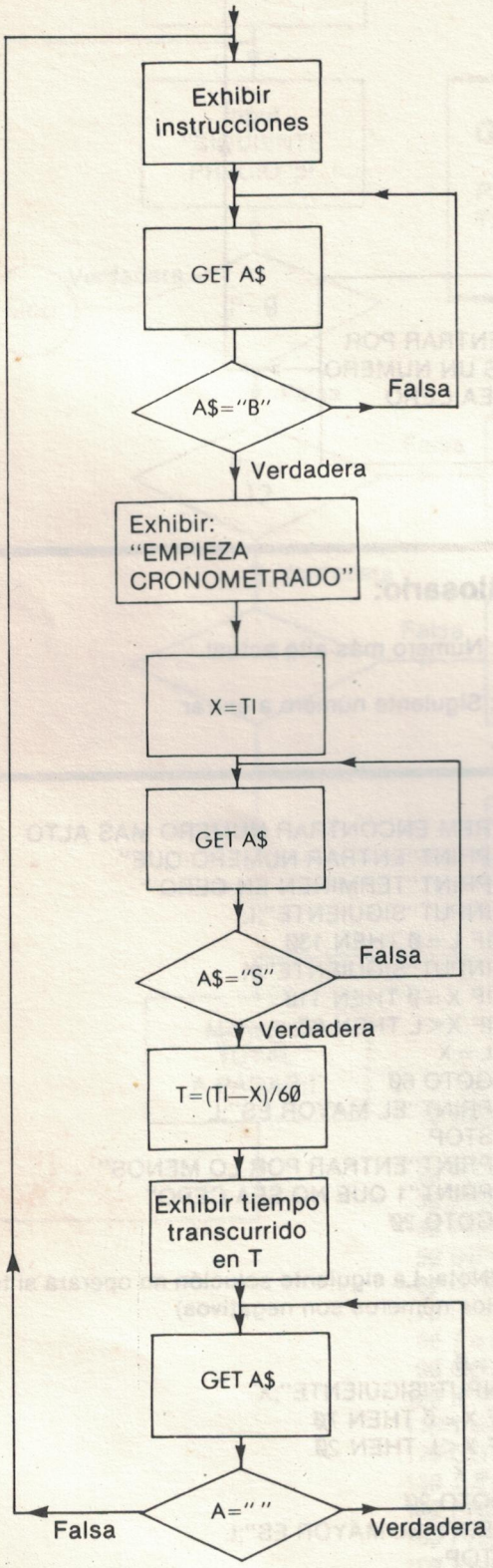
```
1Ø L=Ø
2Ø INPUT"SIGUIENTE";X
3Ø IF X=Ø THEN 7Ø
4Ø IF X<L THEN 2Ø
5Ø L=X
6Ø GOTO 2Ø
7Ø PRINT"EL MAYOR ES";L
8Ø STOP
```

¿Por qué no?

# UNIDAD: 15

Experimento 15.2 (1):



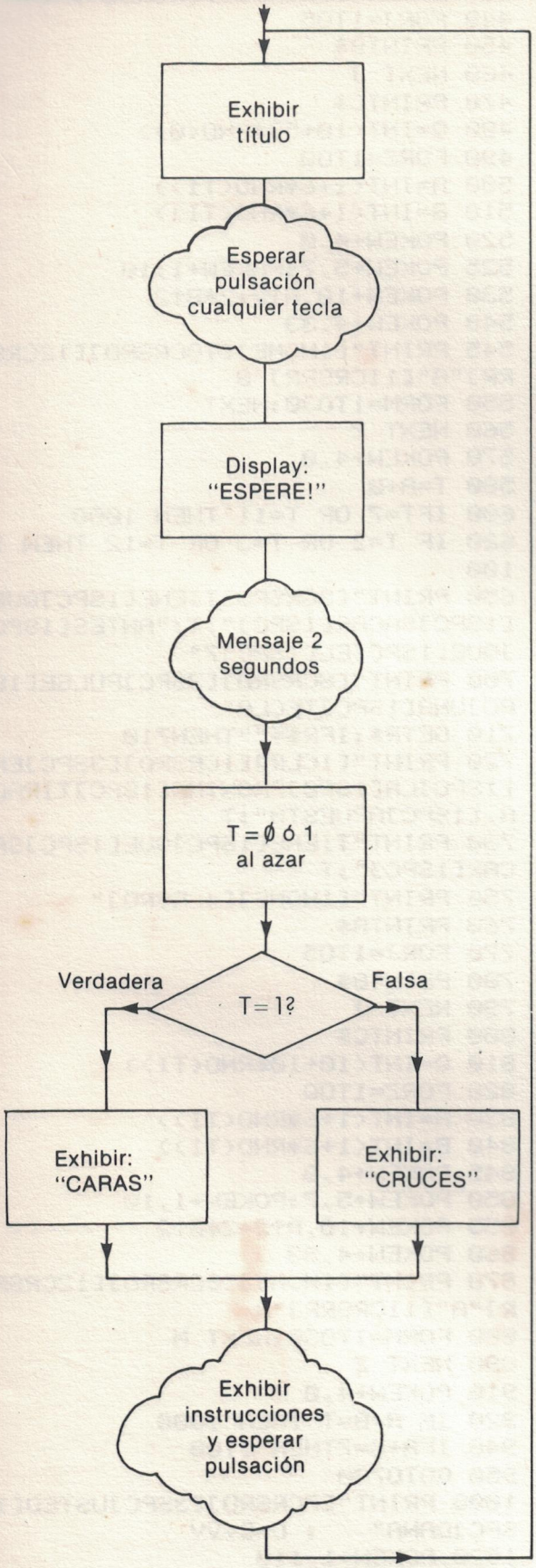



**Glosario:**

A$: Carácter teclado
X: Tiempo interno al inicio intervalo (en 1/60 de segundo)
T: Tiempo transcurrido (segundos)

```
 5 REM CRONOMETRO
 1Ø PRINT" SHIFT y CLR HOME
 2Ø PRINT"PROGRAMA DE CRONOMETRAJE"
 3Ø PRINT
 4Ø PRINT"EMPEZAR CRONOMETRACION"
 5Ø PRINT"PULSANDO TECLA B"
 6Ø PRINT"PARA PARAR, PULSAR S"
 7Ø GET A$
 8Ø IF A$<>"B" THEN 7Ø
 9Ø X=TI
 95 PRINT"CRONOMETRACION EMPEZA-DA"
1ØØ GET A$
11Ø IF A$<>"S" THEN 1ØØ
12Ø T=(TI—X)/6Ø
13Ø PRINT"TIEMPO TRANSCURRIDO"
14Ø PRINT T;"SEGUNDOS"
15Ø PRINT
16Ø PRINT"PULSE CUALQUIER TECLA"
17Ø PRINT"PARA NUEVO CRONOMETRAJE"
18Ø GET A$
19Ø IF A$=" "THEN 18Ø
2ØØ GOTO 1Ø
```

Experimento 15.2 (2):




## Glosario:

S$: Carácter del teclado
M: Usada en bucle para esperar 2 segundos
T: Ø (para cruces) ó 1 (para caras)

```
1Ø REM CARA O CRUZ
2Ø PRINT" SHIFT y CLR HOME "
3Ø PRINT"PULSE CUALQUIER TECLA"
4Ø PRINT"PARA PROBAR SU SUERTE"
5Ø GET S$
6Ø IF S$ = " " THEN 5Ø
7Ø PRINT"ESPERE!"
8Ø PRINT
9Ø FOR M = 1 TO 2ØØØ
1ØØ NEXT M
11Ø T = INT (Ø + 2 ★ RND(Ø))
12Ø IF T = 1 THEN 15Ø
13Ø PRINT"CRUCES"
14Ø GOTO 16Ø
15Ø PRINT"CARAS"
16Ø PRINT
17Ø PRINT"PULSE DE NUEVO"
18Ø PRINT"CUALQUIER TECLA"
19Ø GET S$
2ØØ IF S$ = " "THEN 19Ø
21Ø GOTO 2Ø
```

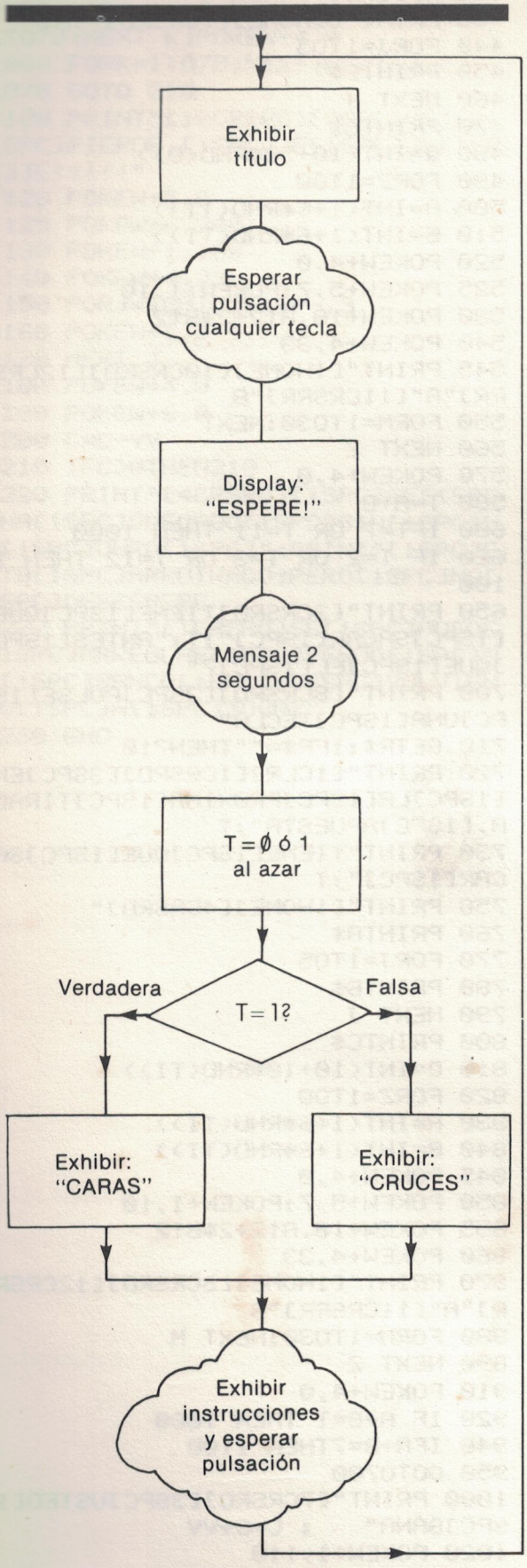


Dream Commodore

Experimento 15.4:

```
5 REM DADOS
10 W=212*256
15 POKEW+24,15
20 POKE53281,0:POKE53280,0
25 POKEW+6,0
30 PRINT"[1CLR][1WHT]"
40 PRINT"[2SPC]EL[1SPC]JUEGO[1SP
C]DE[1SPC]LOS[1SPC]DADOS[1SPC]SE
[1SPC]JUEGA[1SPC]CON"
50 PRINT"[1SPC]DOS[1SPC]DADOS."
60 PRINT"[1SPC]PRIMERO[1SPC]SE[1
SPC]APUESTA[1SPC]Y[1SPC]DESPUES[
1SPC]SE[1SPC]TIRAN[3SPC]LOS[1SPC
]DADOS"
70 PRINT"[1SPC]SI[1SPC]OBTIENE[1
SPC]UN[1SPC][1RVSON]7[1RVSOF][1S
PC]O[1SPC]UN[1SPC][1RVSON]11[1RV
SOF][1SPC]USTED[1SPC]GANA,[1SPC]
SI[1SPC]SALE[1SPC]UN[1SPC][1RVSO
N]2[1CRSRR]3[1RVSOF][1SPC]O[1SPC
]UN[1SPC][1RVSON]12[1RVSOF],[1SP
C]PIERDE"
80 PRINT"[2CRSRD]SI[1SPC]SALE[1S
PC]ALGUN[1SPC]OTRO[1SPC]TIENE[1S
PC]QUE[1SPC]VOLVER[1SPC]A[3SPC]J
UGAR[1SPC]HASTA[1SPC]QUE[1SPC]SA
LGA[1SPC]ALGUNA[1SPC]"
90 PRINT"JUGADA[1SPC]DESCRITA"
100 PRINT"PULSE[1SPC]UNA[1SPC]TE
CLA
110 GETA$:IFA$=""THEN110
120 PRINT"[3CRSRD]CONTINUO"
260 A$="[11SPC][1SHIFU][3SHIF*][
1SHIFI][9SPC][1SHIFU][3SHIF*][1S
HIFI]"
270 B$="[11SPC][1SHIFB][3SPC][1S
HIFB][9SPC][1SHIFB][3SPC][1SHIFB
]"
280 C$="[11SPC][1SHIFJ][3SHIF*][
1SHIFK][9SPC][1SHIFJ][3SHIF*][1S
HIFK]"
285 REM
300 INPUT"[1CLR][5CRSRD]CAPITAL[
1SPC]INICIAL[1SPC]";C
310 PRINT"PULSE[1SPC]UNA[1SPC]TE
CLA"
320 GETV$:IFV$=""THEN320
330 PRINT"[1SPC]SU[1SPC]CAPITAL[
1SPC]AHORA[1SPC]ES[1SPC]DE";C
350 INPUT"[2CRSRD]CUANTO[1SPC]AP
UESTA";VV
360 IFVV<=CTHEN 420
370 PRINT"[1SPC]PUEDE[1SPC]PERMI
TIRSELO"
380 GOTO 350
420 PRINT"[1CLR][3CRSRD][3SPC]PR
IMERA[1SPC]TIRADA"
430 PRINT"[1HOME][7CRSRD]";A$
440 FORJ=1TO5
450 PRINTB$
460 NEXT J
470 PRINTC$
480 Q=INT(10+50*RND(0))
490 FORZ=1TOQ
500 A=INT(1+6*RND(TI))
510 B=INT(1+6*RND(TI))
520 POKEW+4,0
525 POKEW+5,7:POKEW+1,10
530 POKEW+10,A↑2+2*B↑2
540 POKEW+4,33
545 PRINT"[1HOME][10CRSRD][12CRS
RR]"A"[11CRSRR]"B
550 FORM=1TO30:NEXT
560 NEXT Z
570 POKEW+4,0
580 T=A+B
600 IFT=7 OR T=11 THEN 1000
620 IF T=2 OR T=3 OR T=12 THEN 1
100
650 PRINT"[3CRSRD]TIENE[1SPC]QUE
[1SPC]SACAR[1SPC]";T;"ANTES[1SPC
]QUE[1SPC]EL[1SPC]7"
700 PRINT"[8CRSRD][3SPC]PULSE[1S
PC]UNA[1SPC]TECLA"
710 GETR$:IFR$=""THEN710
720 PRINT"[1CLR][1CRSRD][3SPC]EN
[1SPC]LA[1SPC]PROXIMA[1SPC]TIRAD
A,[1SPC]APUESTA";T
730 PRINT"TIENE[1SPC]QUE[1SPC]SA
CAR[1SPC]";T
750 PRINT"[1HOME][4CRSRD]"
760 PRINTA$
770 FORJ=1TO5
780 PRINTB$
790 NEXT J
800 PRINTC$
810 Q=INT(10+10*RND(TI))
820 FORZ=1TOQ
830 A=INT(1+6*RND(TI))
840 B=INT(1+6*RND(TI))
845 POKEW+4,0
850 POKEW+5,7:POKEW+1,10
855 POKEW+10,A↑2+2*B↑2
860 POKEW+4,33
870 PRINT"[1HOME][8CRSRD][12CRSR
R]"A"[11CRSRR]"B
880 FORM=1TO30:NEXT M
890 NEXT Z
910 POKEW+4,0
920 IF A+B=T THEN 1000
940 IFA+B=7THEN 1100
950 GOTO700
1000 PRINT"[7CRSRD][3SPC]USTED[1
SPC]GANA"    : C=C+VV
1020 POKEW+1,110
```

Experimento 15.2 (2):




## Glosario:

S$: Carácter del teclado
M: Usada en bucle para esperar 2 segundos
T: Ø (para cruces) ó 1 (para caras)

```
1Ø REM CARA O CRUZ
2Ø PRINT" SHIFT y CLR HOME "
3Ø PRINT"PULSE CUALQUIER TECLA"
4Ø PRINT"PARA PROBAR SU SUERTE"
5Ø GET S$
6Ø IF S$ = " " THEN 5Ø
7Ø PRINT"ESPERE!"
8Ø PRINT
9Ø FOR M = 1 TO 2ØØØ
1ØØ NEXT M
11Ø T = INT (Ø + 2 ★ RND(Ø))
12Ø IF T = 1 THEN 15Ø
13Ø PRINT"CRUCES"
14Ø GOTO 16Ø
15Ø PRINT"CARAS"
16Ø PRINT
17Ø PRINT"PULSE DE NUEVO"
18Ø PRINT"CUALQUIER TECLA"
19Ø GET S$
2ØØ IF S$ = " "THEN 19Ø
21Ø GOTO 2Ø
```

Experimento 15.4:

```
5 REM DADOS
10 W=212*256
15 POKEW+24,15
20 POKE53281,0:POKE53280,0
25 POKEW+6,0
30 PRINT"[1CLR][1WHT]"
40 PRINT"[2SPC]EL[1SPC]JUEGO[1SP
C]DE[1SPC]LOS[1SPC]DADOS[1SPC]SE
[1SPC]JUEGA[1SPC]CON"
50 PRINT"[1SPC]DOS[1SPC]DADOS."
60 PRINT"[1SPC]PRIMERO[1SPC]SE[1
SPC]APUESTA[1SPC]Y[1SPC]DESPUES[
1SPC]SE[1SPC]TIRAN[3SPC]LOS[1SPC
]DADOS"
70 PRINT"[1SPC]SI[1SPC]OBTIENE[1
SPC]UN[1SPC][1RVSON]7[1RVSOF][1S
PC]O[1SPC]UN[1SPC][1RVSON]11[1RV
SOF][1SPC]USTED[1SPC]GANA,[1SPC]
SI[1SPC]SALE[1SPC]UN[1SPC][1RVSO
N]2[1CRSRR]3[1RVSOF][1SPC]O[1SPC
]UN[1SPC][1RVSON]12[1RVSOF],[1SP
C]PIERDE"
80 PRINT"[2CRSRD]SI[1SPC]SALE[1S
PC]ALGUN[1SPC]OTRO[1SPC]TIENE[1S
PC]QUE[1SPC]VOLVER[1SPC]A[3SPC]J
UGAR[1SPC]HASTA[1SPC]QUE[1SPC]SA
LGA[1SPC]ALGUNA[1SPC]"
90 PRINT"JUGADA[1SPC]DESCRITA"
100 PRINT"PULSE[1SPC]UNA[1SPC]TE
CLA
110 GETA$:IFA$=""THEN110
120 PRINT"[3CRSRD]CONTINUO"
260 A$="[11SPC][1SHIFU][3SHIF*][
1SHIFI][9SPC][1SHIFU][3SHIF*][1S
HIFI]"
270 B$="[11SPC][1SHIFB][3SPC][1S
HIFB][9SPC][1SHIFB][3SPC][1SHIFB
]"
280 C$="[11SPC][1SHIFJ][3SHIF*][
1SHIFK][9SPC][1SHIFJ][3SHIF*][1S
HIFK]"
285 REM
300 INPUT"[1CLR][5CRSRD]CAPITAL[
1SPC]INICIAL[1SPC]";C
310 PRINT"PULSE[1SPC]UNA[1SPC]TE
CLA"
320 GETV$:IFV$=""THEN320
330 PRINT"[1SPC]SU[1SPC]CAPITAL[
1SPC]AHORA[1SPC]ES[1SPC]DE";C
350 INPUT"[2CRSRD]CUANTO[1SPC]AP
UESTA";VV
360 IFVV<=CTHEN 420
370 PRINT"[1SPC]PUEDE[1SPC]PERMI
TIRSELO"
380 GOTO 350
420 PRINT"[1CLR][3CRSRD][3SPC]PR
IMERA[1SPC]TIRADA"
430 PRINT"[1HOME][7CRSRD]";A$
440 FORJ=1TO5
450 PRINTB$
460 NEXT J
470 PRINTC$
480 Q=INT(10+50*RND(0))
490 FORZ=1TOQ
500 A=INT(1+6*RND(TI))
510 B=INT(1+6*RND(TI))
520 POKEW+4,0
525 POKEW+5,7:POKEW+1,10
530 POKEW+10,A↑2+2*B↑2
540 POKEW+4,33
545 PRINT"[1HOME][10CRSRD][12CRS
RR]"A"[11CRSRR]"B
550 FORM=1TO30:NEXT
560 NEXT Z
570 POKEW+4,0
580 T=A+B
600 IFT=7 OR T=11 THEN 1000
620 IF T=2 OR T=3 OR T=12 THEN 1
100
650 PRINT"[3CRSRD]TIENE[1SPC]QUE
[1SPC]SACAR[1SPC]";T;"ANTES[1SPC
]QUE[1SPC]EL[1SPC]7"
700 PRINT"[8CRSRD][3SPC]PULSE[1S
PC]UNA[1SPC]TECLA"
710 GETR$:IFR$=""THEN710
720 PRINT"[1CLR][1CRSRD][3SPC]EN
[1SPC]LA[1SPC]PROXIMA[1SPC]TIRAD
A,[1SPC]APUESTA";T
730 PRINT"TIENE[1SPC]QUE[1SPC]SA
CAR[1SPC]";T
750 PRINT"[1HOME][4CRSRD]"
760 PRINTA$
770 FORJ=1TO5
780 PRINTB$
790 NEXT J
800 PRINTC$
810 Q=INT(10+10*RND(TI))
820 FORZ=1TOQ
830 A=INT(1+6*RND(TI))
840 B=INT(1+6*RND(TI))
845 POKEW+4,0
850 POKEW+5,7:POKEW+1,10
855 POKEW+10,A↑2+2*B↑2
860 POKEW+4,33
870 PRINT"[1HOME][8CRSRD][12CRSR
R]"A"[11CRSRR]"B
880 FORM=1TO30:NEXT M
890 NEXT Z
910 POKEW+4,0
920 IF A+B=T THEN 1000
940 IFA+B=7THEN 1100
950 GOTO700
1000 PRINT"[7CRSRD][3SPC]USTED[1
SPC]GANA"    : C=C+VV
1020 POKEW+1,110
```

```
1030 POKEW+1,8
1040 POKEW+6,0
1050 FORJ=1TO8:POKEW+4,33:FOR K=
1TO70:NEXT K:POKEW+4,0
1060 FORK=1TO70:NEXT:NEXT J
1070 GOTO 310
1100 PRINT"[10CRSRD][3SPC]USTED[
1SPC]PIERDE,[1SPC]JE[1SPC]JE[1SP
C]JE!!!!!"
1120 POKEW+5,0
1125 POKEW+6,240
1130 POKEW+1,100
1140 POKEW+4,33
1150 FORJ=100TO5STEP-.3
1160 POKEW+1,J
1170 NEXT J
1180 POKEW+4,0
1190 POKEW+6,0
1200 C=C-VV
1210 IFC>0THEN310
1220 PRINT"[4CRSRD][1SPC]SE[1SPC
]HA[1SPC]QUEDADO[1SPC]SIN[1SPC]U
N[1SPC]DURO[1SPC]":PRINT"[1SPC]E
STA[1SPC]ARRUINADO;PERO[1SPC]NO[
1SPC]DESESPERE
1222 PRINT"[1CRSRD]SE[1SPC]PUEDE
[1SPC]ARREGLAR[1SPC]YENDO[1SPC]A
L[1SPC]BANCO[1SPC]Y[6SPC]VOLVIEN
DO[1SPC]A[1SPC]JUGAR"
1230 END
```

```
1000 PRINT" CRSR CRSR CRSR CRSR CRSR CRSR CRSR
     VD. GANA"
1010 C=C+W
1020 POKE 36876,15
1030 FOR J=1 TO 20
1040 POKE 36878,240
1050 FOR M=1 TO 25
1060 NEXT M
1070 POKE 36876,0
1080 FOR M=1 TO 25
1085 NEXT M
1090 NEXT J
1093 PRINT" SHIFT y CLR HOME "
1095 GOTO 305
1100 PRINT" CRSR CRSR CRSR CRSR CRSR CRSR CRSR
     VD. PIERDE"
1110 FOR Z=1 TO 2000:NEXT Z
1120 POKE 36878,15
1130 FOR J=220 TO 127 STEP-1
1140 POKE 36874,J
1150 POKE 36875,J
1160 FOR M=1 TO 5
1170 NEXT M
1180 NEXT J
1190 POKE 36878,0
1200 C=C-W
1210 IF C>0 THEN 305
1215 PRINT
1220 PRINT"ESTA SIN BLANCA"
1230 STOP
```



# APENDICE A
## SOLUCION A LOS PROBLEMAS

### Ejemplo 1:

```
10 INPUT V
20 INPUT R
30 PRINT "A="; V↑2/R
40 STOP
```

### Ejemplo 2:

```
10 PRINT "X  FORMULA"
20 FOR X=0 TO 2 STEP 0.2
30 PRINT X; 1/(1+X↑2)
40 NEXT X
50 STOP
```

### Ejemplo 3:

```
 10 PRINT "LONGITUD TRES LADOS"
 20 INPUT "A"; A
 30 INPUT "B"; B
 40 INPUT "C"; C
 50 S= (A+B+C)/2
 60 X= S★(S—A)★(S—B)★(S—C)
 70 IF X <0 THEN 100
 80 PRINT "AREA ES"; SQR(X)
 90 STOP
100 PRINT "NO SON"
110 PRINT "LADOS DE TRIANGULO"
120 STOP
```

**Glosario:**

A, B, C: Tres lados de triángulo
S: Semiperímetro
X: Cuadrado del área (de haberla)

### Ejemplo 4:

```
 10 REM VERSION ALGO MAS RAPIDA
 20 INPUT"VALOR MAS ALTO";H
 30 FOR N=3 TO H STEP 2
 40 Q= SQR (N)
 50 FOR J=2 TO Q
 60 IF N/J= INT (N/J) THEN 90
 70 NEXT J
 80 PRINT N;
 90 NEXT N
100 STOP
```

# APENDICE C



## Mensajes de Error

A continuación se explican los mensajes de error que pueden aparecer usando el vocabulario y sintaxis BASIC tratados en esta Parte I. Pueden surgir otros mensajes de error y señalización de errores si se ejecutan programas de mayor complejidad.

### Division by Zero (División por cero)

No está permitido dividir un número por cero. Este mensaje puede aparecer en instrucciones como las siguientes:

```
10 A = 5/0
```

o

```
20 A = Q/(J—J)
```

### Extra Ignored (Se prescinde de los extras)

Si usted pulsa demasiados elementos (número o cadenas) en respuesta a una instrucción INPUT, el número adicional será omitido. El programa no se interrumpirá.

### Illegal Quantity (Cantidad ilegal)

El número usado en una instrucción es demasiado grande (o demasiado pequeño). Por ejemplo, cualquier número que coloque con POKE en una posición de memoria debe estar comprendido entre 0 y 255.

Este mensaje puede aparecer en instrucciones erróneas del siguiente tipo:

```
10 POKE 36878,1234
```

o

```
20 J = 300
30 POKE 36876,J
```

### Load Error (Error de carga)

El programa no se carga correctamente desde la cinta cassette. Limpie la cabeza de lectura, quizás está sucia. Puede ser también que el programa no esté cargado correctamente o que la cinta se haya dañado por la acción de un campo magnético.

### Next Without For (Next sin For)

La estructura FOR-NEXT del programa está equivocada.

### Out of Memory (Memoria Agotada)

El ordenador ha agotado sus recursos de memoria. Sucede sólo con programas muy largos o que usan grandes cantidades de datos.

### Redo From Start (Empezar de Nuevo)

Si un comando INPUT espera un número y el usuario escribe algo que no sea un número, el ordenador exhibirá este mensaje y esperará a que usted pruebe de nuevo.

### String Too Long (Cadena Demasiado Larga)

Una cadena formada por concatenación tiene más de 255 bytes.

### Syntax Error (Error de Sintaxis)

En una instrucción se infringen las reglas de sintaxis del BASIC. Puede ser la colocación errónea de paréntesis, un error de mecanografía al escribir una palabra clave o inadecuado orden de los elementos de una expresión.

### Type Mismatch (Combinación Errónea de Caracteres)

Significa que se ha usado un número en vez de una cadena o viceversa.

### Verify Error (Error de Verificación)

El proceso de verificación de la grabación de un programa en cinta indica que hay un error. Intente grabar de nuevo el programa.

# APENDICE D



## CLAVE PARA INTERPRETAR LOS LISTADOS DEL CURSO DE INTRODUCCION AL BASIC

TODOS los listados que se publican a continuación han sido ejecutados en el COMMODORE 64, para mejorar su legibilidad por parte del usuario se les ha sometido a ciertas modificaciones mediante un programa escrito especialmente para ello. Para los programas destinados a este ordenador que usa frecuentemente las posibilidades gráficas del teclado— se han sustituido los símbolos gráficos que aparecen normalmente en los listados por una serie de letras entre corchetes ([]) que indican la secuencia de teclas que se deben pulsar para obtener el caracter deseado. A continuación se da una tabla para aclarar la interpretación de las indicaciones entre corchetes:

[CRSRD) = Tecla cursor hacia abajo (sin SHIFT)
[CRSRU] = Tecla cursor hacia arriba (con SHIFT)
[CRSRR] = Tecla cursor a la derecha (sin SHIFT)
[CRSRL] = Tecla cursor a la izquierda (con SHIFT)
[HOME] = Tecla CLR/HOME (sin SHIFT)
[ CLR ] = Tecla CLR/HOME (con SHIFT)
[ SPC ] = Barra espaciadora

Las indicaciones [ BLK ] a [ YEL ] corresponden a la pulsación de las teclas de 1 a 8 junto a la tecla CTRL, lo mismo sucede con [RVSON] y [RVSOF] respecto a la tecla CTRL y las teclas 9 y Ø.

El resto de las indicaciones constan de la parte COMM o SHIF seguidas de una letra, número o símbolo —por ejemplo [COMM+] o [SHIFA]— esto indica que para obtener el gráfico necesario en el programa deben pulsarse simultáneamente las teclas COMMODORE (la que lleva el logotipo) o una de SHIFT y la tecla indicada por la letra, el número o el símbolo, en el ejemplo anterior: COMMODORE y + o SHIFT y A, respectivamente.
Además debe notarse que dentro de los mismos corchetes se incluye un número que indica cuantas veces debe pulsarse la tecla correspondiente, así [16CRSRD] equivale a 16 pulsaciones de la tecla de cursor abajo.

## CARTA DE AJUSTE

```
5 REM ESPACIO DE SPRITES
                X
10 REM ESPACIO DE SPRITES
                X
15 REM ESPACIO DE SPRITES
                X
20 REM ESPACIO DE SPRITES
                X
25 REM ESPACIO DE SPRITES
                X
30 REM ESPACIO DE SPRITES
                X
35 REM ESPACIO DE SPRITES
                X
40 REM ESPACIO DE SPRITES
                X
45 REM ESPACIO DE SPRITES
                X
50 REM ESPACIO DE SPRITES
                X
55 REM ESPACIO DE SPRITES
                X
60 REM ESPACIO DE SPRITES
                X
65 REM ESPACIO DE SPRITES
                X
70 REM ESPACIO DE SPRITES
                X
75 REM ESPACIO DE SPRITES
                X
90 DIM SP(10),N(60),V%(500,3),W%.
(500,3),C$(16)
95 GOTO10210: REM MUSICA CON ENV
OLVENTE CRITICA
100 REM TOCANDO UN TONO<NV VOCES
,KK NOTAS EN PARTE 1
105 FORJ=1TONV:P(J)=0:D(J)=0:T(J
)=TI:NEXTJ
110 FORJ=1TONV
115 IF T(J)>TI THEN 130
```

```
120 T(J)=T(J)+D(J):X=V%(P(J),J)
125 Y=W%(P(J),J):P(J)=P(J)+1:D(J
)=Y*DV:M(J)=N(X)
130 NEXTJ
140 F=0
145 FORJ=1TONV
150 IFT(J)<=TITHEN SP(J+7)=M(J):
F=1
155 NEXTJ
160 IFF=0THEN 145
165 IFP(1)<=KKTHEN110
170 RETURN
500 GOTO5000 :REM PUNTERO DE COD
IGO DE MOVIMIENTO
2000 REM LLUVIA DORADA(1)
2005 SP(7)=A+64080
2010 ONCWGOTO2020,2040,2045,2050

2020 IFRND(0)>0.04THEN 2100
2030 CW=2:CN=50:POKE2040,34:POKE
V+21,PEEK(V+21)+1:POKEV+39,1+7*R
ND(0)
2035 SP(10)=NO:GOTO2100
2040 CW=3:POKE2040,35:GOTO2100
2045 CW=4:POKE2040,36:GOTO2100
2050 CW=2:POKE 2040,34:CN=CN-1
2060 IFCN=0THEN CW=1:POKEV+21,PE
EK(V+21)-1
2100 REM GRANDES EXPLOSIONES
2110 ONGRGOTO2120,2140,2145,2150

2120 IFRND(0)>0.1 THEN 2540
2130 GR=2:POKE2041,37:POKEV+21,P
EEK(V+21)+2:POKEV+40,1+7*RND(0)
2135 SP(10)=NO:GOTO2540
2140 GR=3:POKE2041,38:GOTO2540
2145 GR=4:POKE 2041,39:GOTO2540
2150 GR=1:POKE V+21,PEEK(V+21)-2

2540 REM COHETE(2)
2610 ONQAGOTO2620,2670,2690,2710
,2730
2620 IFRND(0)>0.05ORBA=4 THEN274
0
2630 QY=250:QX=INT(350*RND(0)):Q
V=-19
2640 QH=INT((350-QX-300*RND(0))/
20):POKE2045,33:POKEV+44,1+7*RND
(0)
2650 SP(5)=A+256*QX+QY
2660 POKEV+21,PEEK(V+21)+32:QA=2
:GOTO2740
2670 QX=QX+QH:QY=QY+QV:QV=QV+1
2680 SP(5)=A+256*QX+QY:IFQV=3THE
NQA=3:SP(10)=NO:QN=2:POKE2045,37
:GOTO2740
2690 QN=QN-1:IF QN=0THEN QN=3:QA
=4:POKE2045,38
2700 GOTO2740
2710 QN=QN-1:IFQN<>0THEN GOTO274
0
2715 QN=5:QA=5:POKE2045,39:GOTO2
740
2720 SP(5)=SP(5)-11*257:GOTO2740

2730 QN=QN-1:IFQN<>0THEN 2740
2732 QA=1:POKEV+21,PEEK(V+21)-32

2735 POKEV+23,PEEK(V+23)AND223:P
OKEV+29,PEEK(V+29)AND223
2740 REM GLOBO
2810 ONBAGOTO2820,2850,2870,2882

2820 IF RND(0)>0.02THEN 3000
2830 SP(6)=A+350*256+80:BP=350 :
BA=2:POKE2046,42 :POKEV+23,PEEK(
V+23)+64
2835 POKE V+29,PEEK(V+29)+64
2840 POKEV+45,1+7*RND(0):POKEV+2
1,PEEK(V+21)+64:GOTO3000
2850 BP=BP-1:SP(6)=SP(6)-256
2860 IFBP=0THENPOKEV+21,PEEK(V+2
1)-64:BA=1
2865 IF BP=0THENPOKEV+23,PEEK(V+
23)-64:POKEV+29,PEEK(V+29)-64
2867 GOTO3000
2870 BN=BN-1:IFBN<>0THEN3000
2880 BA=4:POKE2046,41:POKEV+45,1
:BP=80:POKEV+21,PEEK(V+21)+64:GO
TO3000
2882 S1=0:S2=0:IFT1<TQTHENS1=N(V
%(P1,1)):T1=T1+DV*W%(P1,1):P1=P1
+1
2884 IFT2<TQTHEN S2=N(V%(P2,2)):
T2=T2+DV*W%(P2,2):P2=P2+1
2886 SP(8)=S1:SP(9)=S2
2890 BP=BP+1:IFBP=213THENPOKEV+2
1,PEEK(V+21)-64:BA=1
2900 SP(6)=SP(6)+1
3000 P=PEEK(V+30)AND96:IF(P<>96)
OR(BA<>2)OR(QA<>2)THEN 3500
3010 REM COLISION COHETE Y GLOBO

3015 P1=0:P2=0:T1=TQ+5:T2=T1
3020 QA=3:QN=2:POKE2045,39:SP(10
)=NO:BN=8:BA=3:POKEV+21,PEEK(V+2
1)-64
3030 SP(6)=SP(6)+1028:POKE V+23,
PEEK(V+23)-64:POKEV+29,PEEK(V+29
)-64:GOTO2000
3500 TQ=TQ+1:POKE V+21,PEEK(V+21
)OR128:IFTQ<500 OR  BA<>1THEN 20
00
3505 FORJ=1TO1000:NEXTJ
3510 FORJ=0TO10:SP(J)=0:POKEVV+2
4,0
3515 POKEV+21,0
3520 PRINT"[1CLR][4CRSRD][5SPC]F
INALMENTE[1SPC]DOS[1SPC]PIEZAS[2
SPC]"
```

```
3525 PRINT
3530 PRINT"[5SPC]POR[2SPC]WOLFGA
NG[1SPC]AMADEUS
3540 PRINT"[1CRSRD][11SPC]M[1SPC
]O[1SPC]Z[1SPC]A[1SPC]R[1SPC]T
3550 PRINT:PRINT"[2SPC]PRIMERO,[
3SPC]ANDANTE"
3560 GOSUB 5040:GOSUB100
3570 PRINT:PRINT"[2SPC]DESPUES,[
1SPC]UN[2SPC]ALLEGRO[1SPC]"
3580 GOSUB 5040:GOSUB100
3590 RESTORE:
3600 PRINT:PRINT "EL[1SPC]PROGRA
MA[1SPC]SERA[1SPC]REPETIDO[1SPC]
AHORA"
3700 FORJ=832TO1000:READA:NEXTJ
3710 POKEVV+4,0:POKEVV+11,0:POKE
VV+18,0:POKEVV+24,0
3720 GOTO10290
4000 C$(1)="[3SPC][1BLK][1RVSON]
[15SPC][1RVSOF][1WHT][1SPC]NEGRO
"
4010 C$(2)="[3SPC][1WHT][1RVSON]
[15SPC][1RVSOF][1WHT][1SPC]BLANC
O"
4030 C$(3)="[3SPC][1RED][1RVSON]
[15SPC][1RVSOF][1WHT][1SPC]ROJO"

4040 C$(4)="[3SPC][1CYN][1RVSON]
[15SPC][1RVSOF][1WHT][1SPC]CIAN"

4050 C$(5)="[3SPC][1PUR][1RVSON]
[15SPC][1RVSOF][1WHT][1SPC]PURPU
RA"
4060 C$(6)="[3SPC][1GRN][1RVSON]
[15SPC][1RVSOF][1WHT][1SPC]VERDE
"
4070 C$(7)="[3SPC][1BLU][1RVSON]
[15SPC][1RVSOF][1WHT][1SPC]AZUL"

4080 C$(8)="[3SPC][1YEL][1RVSON]
[15SPC][1RVSOF][1WHT][1SPC]AMARI
LLO"
4090 C$(9)="[3SPC][1COMM1][1RVSO
N][15SPC][1RVSOF][1WHT][1SPC]NAR
ANJA"
4100 C$(10)="[3SPC][1COMM2][1RVS
ON][15SPC][1RVSOF][1WHT][1SPC]MA
RRON"
4110 C$(11)="[3SPC][1COMM3][1RVS
ON][15SPC][1RVSOF][1WHT][1SPC]RO
JO[1SPC]CLARO"
4120 C$(12)="[3SPC][1COMM4][1RVS
ON][15SPC][1RVSOF][1WHT][1SPC]GR
IS[1SPC]OSCURO"
4130 C$(13)="[3SPC][1COMM5][1RVS
ON][15SPC][1RVSOF][1WHT][1SPC]GR
IS[1SPC]MEDIO"
4140 C$(14)="[3SPC][1COMM6][1RVS
ON][15SPC][1RVSOF][1WHT][1SPC]VE
RDE[1SPC]CLARO"
4150 C$(15)="[3SPC][1COMM7][1RVS
ON][15SPC][1RVSOF][1WHT][1SPC]AZ
UL[1SPC]CLARO"
4160 C$(16)="[3SPC][1COMM8][1RVS
ON][15SPC][1RVSOF][1WHT][1SPC]GR
IS[1SPC]CLARO"
4170 POKE 53280,1:POKE53281,6:PO
KE208*256+21,0
4180 PRINT"[1CLR][1WHT][1CRSRD][
1SPC]!HOLA![1SPC]BIENVENIDO[1SPC
]AL[1SPC][1RVSON]COMMODORE[1SPC]
64[1RVSOF]"
4190 PRINT
4200 PRINT"[1SPC]HE[1SPC]AQUI[1S
PC]UNOS[1SPC]EJEMPLOS[1SPC]DE[1S
PC]COLORES[1SPC]PARA
4210 PRINT"[4SPC]AYUDAR[1SPC]A[1
SPC]AJUSTAR[1SPC]SU[1SPC]TELEVIS
OR"
4220 PRINT
4230 GOTO10300
5000 REM CALCULO DE LAS NOTAS
5010 A=2↑31+33*2↑16:Q=720:P=2↑(1
/12)
5020 FORJJ=1TO60: N(JJ)=A+Q:Q=Q*
P:NEXTJJ
5030 REM ENTRAR AQUI SI N OTRA V
EZ CALCULADA
5040 VV=212*256:N(5)=A
5050 READ NV,NS,AT,D,S,DV
5060 FORJJ=1TONV:PP(JJ)=0:NEXTJJ

5065 FORKK=1TONS
5070 READDN
5075 IFDN>1 THENFORJJ=1TONV:PQ(J
J)=PP(JJ):NEXTJJ
5078 FORJJ=1TONV
5080 READA$:READB$
5090 FORTT=1TOLEN(A$)
5100 V%(PP(JJ),JJ)=ASC(MID$(A$,T
T,1))-35
5110 W%(PP(JJ),JJ)=ASC(MID$(B$,T
T,1))-48:PP(JJ)=PP(JJ)+1
5120 NEXTTT,JJ
5121 IF DN=1THEN 5129 :REM REPIT
O COPIA
5122 FORJJ=1TONV
5123 PR=PP(JJ)
5124 V%(PP(JJ),JJ)=V%(PQ(JJ),JJ)
:W%(PP(JJ),JJ)=W%(PQ(JJ),JJ)
5125 PQ(JJ)=PQ(JJ)+1:PP(JJ)=PP(J
J)+1
5126 IF PQ(JJ)<PR THEN 5124
5127 NEXT JJ
5129 NEXTKK
5130 KK=PP(1)-1
5140 FORJJ=0TONV-1
5150 POKEVV+7*JJ+5,16*AT+D
5160 POKE VV+7*JJ+6,16*S
```

```
5170 NEXTJJ
5180 POKEVV+24,15
5190 RETURN
6100 REM RUTINA CODIGO MAQUINE P
ARA UTILIZAAR SPRITES Y GENERADO
R SONIDO
6110 REM
6120 DATA165,47,133,251,165,48,1
33,252,160,0,177,251,201,83,208,
8,200,177,251
6130 DATA201,80,240,38,136,200,2
00,177,251,133,253,200,177,251,1
33,254,24,165,251
6140 DATA101,253,133,251,165,252
,101,254,133,252,197,50,208,212,
165,251
6150 DATA197,49,208,206,76,49,23
4
6160 DATA152,24,105,6,168,162,0,
200,200,24,177,251,41,1,240,1,56
,102,255,200
6170 DATA177,251  ,157,0,208,200
,177,251,157,1,208,232,232,200,2
24,16,208,225
6180 DATA165,255,141,  16,208
6182 DATA162,0,200,200,177,251,2
40,29,169,0,157,4,212,177,251,15
7,4,212,169,0
6184 DATA145,251,200,177,251,157
,1,212,200,177,251,157,0,212,76,
208,3
6186 DATA200,200,200,138,24,105,
7,170,224,21,208,209,76,49,234
6190 REM INSERTAR CODIGO EN 988
6200 DATA120,169,64,141,20,3,169
,3,141,21,3,88,96
6205 REM SPRITE DATO- PUNTO BRIL
LANTE
6210 DATA0,0,0,0,0,0,0,0,0
6215 DATA0,0,0,0,0,0,0,0,0
6220 DATA0,0,0,0,0,0,0,24,0
6225 DATA0,60,0,0,60,0,0,24,0
6230 DATA0,0,0,0,0,0,0,0,0
6235 DATA0,0,0,0,0,0,0,0,0
6240 DATA0,0,0,0,0,0,0,0,0
6300 REM LLUVIA DORADA(1)
6310 DATA0,32,0,8,132,136,18,0,2

6315 DATA32,162,164,9,16,146,48,
105,40
6320 DATA64,146,2,0,170,160,148,
100,0
6325 DATA0,90,2,72,36,144,0,124,
8
6330 DATA66,124,32,0,124,0,144,1
24,4
6335 DATA66,124,136,0,124,0,32,1
24,36
6340 DATA0,124,0,77,125,18,0,124
,0
6350 REM LLUVIA DORADA(2)
6360 DATA0,64,0,17,2,68,36,0,1
6365 DATA65,33,82,16,0,0,33,105,
8
6370 DATA16,146,32,1,18,2,4,36,1
60
6375 DATA144,72,0,0,36,2,72,124,
144
6380 DATA0,124,8,66,124,33,0,124
,0
6385 DATA144,124,4,66,124,136,32
,124,0
6390 DATA0,124,4,5,124,0,72,125,
18
6400 REM LLUVIA DORADA(3)
6410 DATA0,128,0,34,1,34,72,64,4

6415 DATA130,16,161,16,128,4,2,3
6,128
6420 DATA16,146,8,32,8,32,2,164,
2
6425 DATA4,72,160,144,36,0,0,124
,2
6430 DATA72,124,144,0,124,8,66,1
24,33
6440 DATA0,124,0,144,124,4,66,12
4,136
6445 DATA32,124,0,0,124,4,5,124,
0
6450 REM EXPLOSION(1)
6455 DATA0,0,0,0,0,0,0,0,0
6460 DATA0,0,0,0,0,0,0,0,0
6465 DATA0,0,0,0,56,0,0,198,0
6470 DATA1,147,0,3,73,128,2,16,1
92
6475 DATA3,66,192,1,137,128,0,19
5,0
6480 DATA0,124,0,0,0,0,0,0,0
6485 DATA0,0,0,0,0,0,0,0,0
6500 REM EXPLOSION(2)
6505 DATA0,0,0,0,0,0,0,0,0
6510 DATA0,0,0,0,16,0,0,16,128
6515 DATA8,57,0,4,198,0,3,33,128

6520 DATA6,0,192,12,72,224,12,4,
120
6525 DATA62,0,96,6,16,192,3,36,1
92
6530 DATA5,129,160,8,254,16,0,48
,0
6535 DATA0,32,0,0,32,0,0,0,0
6550 REM EXPLOSION(3)
6555 DATA0,16,0,0,16,0,0,16,16
6560 DATA73,16,32,32,17,64,18,20
,128
6565 DATA8,17,64,68,2,16,2,84,0
6570 DATA17,9,8,32,0,32,0,72,127

6575 DATA254,0,0,0,2,4,18,32,72
6580 DATA4,36,32,8,160,144,18,32
```

```
,8
6585 DATA32,33,4,64,32,2,0,32,0
6600 REM LUNA
6605 DATA252,0,0,63,240,0,15,252
,0
6610 DATA7,255,128,3,255,240,1,2
55,248
6615 DATA0,255,252,0,127,254,0,6
3,254
6620 DATA0,63,255,0,63,255,0,63,
255
6625 DATA0,63,255,0,127,254,0,12
7,254
6630 DATA0,255,252,1,255,248,3,2
55,224
6635 DATA7,255,128,31,248,0,254,
0,0
6650 REM PARACAIDISTA
6655 DATA0,254,0,15,255,224,62,2
51,248
6660 DATA126,250,252,254,26,254,
254,216,126
6665 DATA126,30,252,127,255,252,
32,0,8
6670 DATA16,0,16,8,0,32,4,0,64
6675 DATA2,0,128,1,1,0,0,130,0
6680 DATA0,84,0,0,124,0,0,56,0
6685 DATA0,40,0,0,108,0,0,0,0
6700 REM GLOBO COMMODORE
6705 DATA0,127,0,1,255,192,3,255
,224
6710 DATA3,231,224,7,217,240,7,2
23,240
6715 DATA7,217,240,3,231,224,3,2
55,224
6720 DATA3,255,224,2,255,160,1,1
27,64
6725 DATA1,62,64,0,156,128,0,156
,128
6730 DATA0,73,0,0,73,0,0,62,0
6735 DATA0,62,0,0,62,0,0,28,0
7500 REM GAVOTA DE G. F. HAENDEL

7502 DATA3,2,0,10,0,15
7505 DATA1
7510 DATA"(ONOQOJSQSTSQOVTSQOQON
OQOJSQSTSQOVTSQOQ"
7520 DATA"811112211112222222411
11221111222222224"
7530 DATA"((JJLOLNOLIJ(JJLOLNOLI
J"
7540 DATA"884422112224844221122
4"
7550 DATA"(7;>CBCECB@GEC<@><;97;
>CBCECB@GEC<@>"
7560 DATA"842211112222222211114
21111222222224"
7565 DATA2
7570 DATA"JHGLHGEJGECHGECBCBCEC>
HGHJGEGHEJCBCEGEC"
7580 DATA"211421142112222411112
1111111221111224"
7590 DATA"CGECE(ECBC(CC@>(((CCBC
ECB;"
7600 DATA"421122211222222886211
1224"
7610 DATA";7496427;479;><;9;79BC
7><;@<>27"
7620 DATA"42242112222222222222
211222224"
8000 REM MARCHA MARINES USA
8010 DATA2,2,0,12,0,1
8015 DATA1
8020 DATA"(AEHHHHHMHEFHHFCAAEHHH
HHMHEFHHFCAML"
8030 DATA"81122223121122226112222
2312112222611"
8040 DATA"(5599(9(9(9(9999(44(4(
4(459999(99(9(9(9(9999(44(4(4(45
9999("
8050 DATA"81111111111111122111111
1111111221111111111111221111111111
11122"
8055 DATA1
8060 DATA"JFJFHJHMLJFJMHAEHHHHHM
HEFHHFCA"
8070 DATA"22223121122226112222312
1122227"
8080 DATA"(:(5(:(5(9999((:(5(:(5
(9999(99(9(9(9(9999(44(4(4(45999
9"
8090 DATA"11111111111122111111111
1112211111111111112211111111111
3"
8500 REM ANDANTE
8510 DATA2,2,3,11,0,17
8515 DATA2
8520 DATA"(MOMOQMHHHOQOQROHHHTLM
OLMVSOMLTQVTSST("
8530 DATA"441111222241111222241
1141111111122422"
8540 DATA"(EHEAEHEHEAHMLMOLHLOLO
LHE(HJGHMOOOH("
8550 DATA"444111111114111111111
11224442222422"
8595 DATA2
8600 DATA"TQRTVROQRTRQOQMRQO(RQO
MLTRQOMJVROMLLM("
8610 DATA"411114111111112242241
1141111111122422"
8620 DATA"(QNJ(OLHMLJOMAEAH(OLH(
EMJ(GH((A("
8630 DATA"22222222222111124222
2223122422"
8700 REM ALLEGRO
8710 DATA  2,2,1,9, 0,10
8715 DATA2
8720 DATA"(QTRQOM(MOOOOQM(MOOOOQ
MOQRTVXYVQST(TVXYVSTVSTVXYTXVTVX
YVSTVST("
```

```
8730 DATA"8222224222221142222211
21111112222422111111111111111111
11111142"
8740 DATA"(MQOMHE(EHHHHE(EHHHHE(
MLOLOMOMOLOLOMOMOL("
8750 DATA"8222224222222422222244
82222222222222222242"
8795 DATA2
8800 DATA"HTQVTROTRQMRQOMT(QTRQO
MOQRTVXYXMOQRTQOMXY(T(R(O(M(HMMM
HQQQMTOQM("
8810 DATA"26262626264444242222222
11111111811111111822222224222222
2222243142"
8820 DATA"(AMJOMLHMLKJHFGH(MQOMH
E(QROTRQ(QROTRQ(Q(O(H(E((AAA(EEE
AH<A("
8830 DATA42626262624444442222224
22222242222222222222242222222222
24442"
10210 FORJ=832 TO 1000
10220 READA:POKEJ,A:NEXTJ
10230 SYS(988)
10240 REM HACIENDO ESPACIO PARA
SPRITES
10250 A=PEEK(43)+256*PEEK(44)
10260 IF A> 2048+14*64THEN 10290

10270 POKE 43,PEEK(A):POKE 44,PE
EK(A+1):GOTO10250
10290 GOTO4000
10300 FOR J=1TO10:PRINTC$(J):  Q
0=2048+64*J:GOSUB11000:NEXTJ
10305 FORJ=11TO16:PRINT C$(J):FO
RK=1TO 200:NEXTK,J
10310 V=208*256
10312 PRINT"[1CRSRD][1SPC]AHORA[
1SPC]SUBA[1SPC]EL[1SPC]VOLUMEN[1
SPC].....[1SPC]Y[1SPC]ESPERE"
10315 GOSUB 5000:GOSUB100
10318 FORK=1TO2000:NEXTK
10320 POKE53280,0:POKE 53281,0:P
RINT"[1CLR]"
10330 POKEVV+24,0
10340 PRINT "[1CLR][2SPC][1COMM8
][2SPC]FALLAS[1SPC]NOCTURNAS...Y
[1SPC]AQUI[1SPC]ESTA"
10350 PRINT"[1SPC]UNA[1SPC]DEMOS
TRACION[1SPC]DE[1SPC]FUEGOS[1SPC
]ARTIFICIALES[5SPC]PARA[1SPC]UST
ED"
10360 FOR J=1TO 30:POKE 1134+460
*RND(0),46:FORK=1TO100:NEXTK,J
10400 REM
10405 SP(8)=2↑31+2↑16:SP(9)=2↑31
+2↑16
10410 A=2↑31:VV=212*256:POKEVV+1
9,12:POKEVV+20,0
10420 SP(0)=A+50*256+200:SP(1)=A
+280*256+180
10430 SP(7)=A+250*256+80:POKEV+4
6,7:POKE 2047,40:POKEV+21,128
10440 NO=2↑31+129*2↑16+5000:POKE
VV+24,15
10450 CW=1:GR=1:QA=1:BA=1:GOSUB5
040:TQ=0
10460 POKE 2042,33:POKE2043,33:
10470 POKE V+23, 3:POKE V+29, 3
10480 GOTO 2000
11000 REM COLOCO UN SPRITE EN LO
CALIZACION Q0
11010 FORJJ=0TO62
11020 READ KK
11030 POKEQ0+JJ,KK
11040 NEXTJJ
11050 RETURN
```

PROGRAMA: AHORCADO

```
0 REM AHORCADO 64
5 REM DERECHOS DE COPIA (C) ANDR
EW COLIN 1982
7 REM GRAFICOS POR JOHN COLIN
10 REM AHORCADO 64
15 V=212*256:FOR J=0TO14STEP 7:P
OKE V+J,0:POKEV+J+4,0:POKE V+J+5
,10
17 POKE V+J+6,0:NEXTJ
18 POKEV+24,15
20 IF Z>13 THEN 220
25 POKE 53281,1:PRINT"[1CLR]"
30 DIM P%(240),Q%(240),R%(240),W
$(500)
35 RESTORE
70 FORJ=0TO240:READP%(J):NEXTJ
72 FORJ=0TO240:READQ%(J):NEXTJ
74 FORJ=0TO240:READR%(J):NEXTJ
80 FORJ=1TO6:FORK=1TO4:READB%(J,
K):NEXTK,J
85 W=0
90 READ W$(W):IF W$(W)<>"ZZZZ" T
HEN W=W+1:GOTO90
95 K=TI:J=K-100*INT(K /100)
97 FORK=0TOJ:L=RND(0):NEXTK
100 Q%(109)=202:Q%(111)=203:FORJ
=98TO115:R%(J)=10:NEXTJ:GOSUB120
0
110 PRINT"[1CLR]"
120 X=23:Y=0:A=0:B=89:R=LEN(X$)
130 GOSUB1000
135 P=23:Q=0
140 PRINT"[1HOME]PALABRA[1SPC]DE
";R;"[1SPC]LETRAS"
150 FORJ=1TOR:POKE1148+J,42:POKE
55420+J,6:NEXTJ
160 PRINT"[1HOME][16CRSRD]LETRAS
":PRINT:PRINT"[1SPC]USADAS":PRIN
T:PRINT"PROXIMA?"
165 T$=MID$("********",1,R)
166 K=0
170 GETZ$:IFZ$=""THEN170
180 Z=ASC(Z$)-64
190 IFZ<1ORZ>26THEN170
200 POKE1916,Z:POKE56188,2
205 IF Z>13 THEN 220
210 POKE1105+40*(Z-1),Z:POKE5537
7+40*(Z-1),0:GOTO240
220 POKE1107+40*(Z-14),Z:POKE553
79+40*(Z-14),0
240 GOSUB1300
250 IFL=1THEN300
260 GOSUB1400
270 K=K+1
280 IFK<>7THENGOSUB1100:GOTO170
290 GOTO500
300 IFT$=X$THEN600
305 GOSUB1500
310 GOTO170
500 GOSUB1700
510 Q%(109)=213:Q%(111)=201
520 FORJ=98TO115:R%(J)=14:NEXTJ
540 FORJ=0TO5
550 POKE1087+40*J,66:POKE55359+4
0*J,0
551 NEXTJ
560 P=23:Q=3
565 FORK=1TO6:GOSUB1100:NEXT
570 PRINT"[1HOME][1BLK]PERDIO.ER
A[13SPC]":PRINT"[3SPC]";X$
580 PRINT"[14SPC]":PRINT"[3SPC]A
HORA[1SPC]PULSE":PRINT"[5SPC]UNA
[1SPC]TECLA":POKEV+5,9
590 A=230:B=70:GOTO630
600 X=23:Y=0:A=0:B=89
610 GOSUB1700
615 FORK=1TO6:GOSUB1100:NEXT
620 PRINT"[1HOME]HA[1SPC]GANADO[
1SPC]ESTA[1SPC]LIBRE.":PRINT"[4S
PC]PULSE[1SPC]UNA[1SPC]TECLA"
625 A=200:B=50
630 POKE V+1,70
640 POKEV+4,0:POKEV+4,33
670 FORM=1TOB
680 GETZ$:IFZ$<>""THENPOKE V+1,0
:POKEV+4,0:GOTO100
690 NEXTM
700 GOTO640
1000 F=40*Y+X
1010 F=40*Y+X
1020 FORJ=ATOB
1040 POKE F+P%(J)+1024,Q%(J)
1050 POKE F+P%(J)+55296,R%(J)
1060 NEXTJ
1070 RETURN
1100 X=P+B%(K,1):Y=Q+B%(K,2)
1110 X=P+B%(K,1):Y=Q+B%(K,2)
1120 A=B%(K,3):B=B%(K,4)
1130 GOSUB 1000
1140 RETURN
1200 X$=W$(INT((W-1)*RND(0)))
1210 RETURN
1300 REM COMPARACION LETRA CON X
$
1305 L=0
1310 FORM=1TOR
1320 IFZ$<>MID$(X$,M,1)THEN1360
1330 T$=LEFT$(T$,M-1)+Z$+RIGHT$(
T$,R-M)
1340 POKE1148+M,Z:POKE55420+M,0
1350 L=1
1360 NEXTM
1370 RETURN
1400 REM SONIDO FALLO
1402 POKE V+5,15
```

```
1405 POKEV+4,0:POKEV+4,33
1410 FORL=50TO  3 STEP-1
1415 POKE V+1,L
1440 NEXTL
1450 POKEV+4,0:RETURN
1500 REM SONIDO EXITO
1505 POKE V+5,9:POKE V+12,9:POKE
 V+19,9
1510 POKEV+4,0:POKEV+1,40:POKEV+
4,33
1520 FORL=0TO100:NEXTL
1530 POKEV+11,0:POKEV+8,50:POKEV
+11,33
1540 FORL=0TO100:NEXTL
1550 POKE V+18;0:POKEV+15,60:POK
E V+18,33
1560 FORL=1TO200:NEXTL
1570 POKE V+1,0:POKEV+8 ,0:POKEV
+15,0:RETURN
1700 REM BORRAR HOMBRE
1710 PRINT"[1HOME]"
1720 PRINT"[13CRSRR][11SPC]"
1730 PRINT"[13CRSRR][12SPC]"
1740 PRINT"[13CRSRR][14SPC]"
1745 PRINT"[13CRSRR][15SPC]"
1747 PRINT"[13CRSRR][16SPC]"
1750 FORJ=6TO21:PRINT"[13CRSRR][
18SPC]"
1760 NEXTJ
1770 RETURN
4000 REM HORCA
4010 DATA0,1,2,3,4,5,6,7,8,9,10
4020 DATA40,43,44,49,50,84,85,89
,90,125,126,129,130
4030 DATA166,167,169,170,207,208
,209,210,248,249,250
4040 DATA289,290,329,330,369,370
,409,410,449,450,489,490
4050 DATA529,530,569,570,609,610
,649,650,689,690,729,730
4060 DATA769,770,808,809,810,811
,848,849,850,851
4070 DATA876,877,878,879,880,881
,882,883,884,885,886,887
4080 DATA888,889,890,891,892,893
,894,895,896
4100 REM CABEZA (P%)
4110 DATA0,37,38,39,40,41,42,43
4120 DATA78,79,80,81,82,118,119,
120,121,122
4130 DATA158,159,160,161,162,199
,200,201
4200 REM CUERPO(P%)
4210 DATA0,1,2,3,4,5,6,40,41,42,
43,44,45,46
4220 DATA80,81,82,83,84,85,86,12
0,121,122,123,124,125,126
4230 DATA160,161,162,163,164,165
,166,200,201,202,203,204,205,206
4240 DATA 240,241,242,243,244,24
5,246
4300 REM BRAZO IZQUIERDO(P%)
4310 DATA0,1,39,40,41,79,80,119,
120,159,160,199,200,240
4400 REM BRAZO DERECHO(P%)
4410 DATA0,1,40,41,42,81,82,121,
122,161,162,201,202,241
4500 REM PIERNA IZQUIERDA(P%)
4510 DATA0,1,2,40,41,42,80,81,82
,120,121,122,160,161,162,200,201
,202
4520 DATA239,240,241,279,280,281

4600 REM PIERNA DERECHA(P%)
4610 DATA0,1,2,40,41,42,80,81,82
,120,121,122,160,161,162,200,201
,202
4620 DATA241,242,243,281,282,283

5000 REM HORCA(Q%)
5010 DATA160,160,160,160,160,160
,160,160,160,160,160,160
5020 DATA66,95,223,160,160,95,22
3,160,160,95,223,160,160
5030 DATA95,223,160,160,95,223,1
60,160,95,160,160
5040 DATA160,160,160,160,160,160
,160,160,160,160,160,160
5050 DATA160,160,160,160,160,160
,160,160,160,160,160,160
5060 DATA160,160,160,160,160,160
,160,160,160,160
5070 DATA104,104,104,104,104,104
,104,104,104,104,104,104
5080 DATA160,160,160,160,104,104
,104,104,104
5100 REM CABEZA (Q%)
5110 DATA66,111,233,160,160,160,
223,111
5120 DATA160,209,160,209,160,160
,160,241,160,160
5130 DATA95,202,195,203,105,160,
160,160
5200 REM CUERPO (Q%)
5210 DATA160,227,160,160,160,227
,160,160,160,160,102,160,160,160

5220 DATA160,160,209,102,167,160
,160,160,160,209,102,167,160,169

5230 DATA160,160,209,102,172,160
,160,163,163,163,163,163,163,163

5240 DATA244,160,160,157,160,160
,231
5300 REM BRAZO IZQUIERDO(Q%)
5310 DATA233,160,106,160,160,106
,160,106,160,106,160,106,160,160
```

```
5400 REM BRAZO DERECHO(Q%)
5410 DATA160,223,160,160,116,160
,116,160,116,160,116,160,116,160

5500 REM PIERNA IZQUIERDA(Q%)
5510 DATA160,160,160,160,160,160
,160,160,160,160,160,160,160,160,160
,160,160
5520 DATA160,160,98,98,160,120,1
20,120
5600 REM PIERNA DERECHA(Q%)
5610 DATA160,160,160,160,160,160
,160,160,160,160,160,160,160,160,160
,160,160
5620 DATA160,160,160,98,98,120,1
20,120
6000 REM HORCA (R%)
6010 DATA0,0,0,0,0,0,0,0,0,0,0,0
6020 DATA0,0,0,0,0,0,0,0,0,0,0,0,0
,0
6030 DATA0,0,0,0,0,0,0,0,0,0,0,0
6040 DATA0,0,0,0,0,0,0,0,0,0,0,0,0

6050 DATA0,0,0,0,0,0,0,0,0,0,0,0

6060 DATA0,0,0,0,0,0,0,0,0,0,0
6070 DATA5,5,5,5,5,5,5,5,5,5,5

6080 DATA0,0,0,0,5,5,5,5,5
6100 REM CABEZA (R%)
6110 DATA0,0,0,0,0,0,0,0
6120 DATA10,10,10,10,10,10,10,10
,10,10
6130 DATA10,10,10,10,10,10,10,10

6200 REM CUERPO(R%)
6210 DATA0,0,0,2,0,0,0,0,0,0,2,0
,0,0
6220 DATA0,0,0,2,0,0,0,0,0,0,2,0
,0,0
6230 DATA0,0,0,2,0,0,0,4,4,4,4,4
,4,4
6240 DATA4,4,4,4,4,4,4
6300 REM BRAZO IZQUIERDO(R%)
6310 DATA0,0,0,0,0,0,0,0,0,0,0,0,0
,0,10
6400 REM BRAZO DERECHO(R%)
6410 DATA0,0,0,0,0,0,0,0,0,0,0,0,0
,0,10
6500 REM PIERNA IZQUIERDA(R%)
6510 DATA4,4,4,4,4,4,4,4,4,4,4,4
,4,4,4,4,4,4,0,0,6,0,0,0
6600 REM PIERNA DERECHA(R%)
6610 DATA4,4,4,4,4,4,4,4,4,4,4,4
,4,4,4,4,4,4,6,0,0,0,0,0
6900 DATA0,2,90,115,-3,7,116,164
,-5,7,165,178,4,7,179,192,-3,14,
193,216
6910 DATA1,14,217,240
7000 DATAHOJA,LIBRO,PAPEL,RELOJ,
MOTOCICLETA,COCHE,AVION,ORDENADO
R,PAJARO
7010 DATAJUEZ,MUJER,HOMBRE,MARCH
A,CAMBIO,AJO,HIJO,OJO,JERINGA,LL
UVIA,LECHE
7020 DATACARPETA,RUEGO,LLAMADA,T
ELEFONO,DISCO,COMMODORE
7030 DATAEMPRESA,TEXTO,PALABRA,L
APIZ,BOLIGRAFO,PLUMA,DINERO,CENA

7040 DATACURSO,LECCION,TRAMPA,LO
CO,LISTO,NARANJA,LIMON,AZUL
7050 DATA LLAVE,PERRO,GATO,CABAL
LO,PEZ,BASICO,PROGRAMA,COMITE,PI
ERNA,BRAZO
7060 DATA CLAVE,LISTA,FILA,COLUM
NA,CANTIDAD,BARCELONA,FIGURA,ZZZ
Z
```

PROGRAMA: VELMEC

```
1 REM VELMEC 64
10 V=212*256:POKEV+24,15:POKEV+5
,8:POKEV+6,0
20 POKE 53281,11:PRINT"[1COMM8][
1CLR]"
30 PRINT"[1CLR][1CRSRD][4SPC]!HO
LA![1SPC]ESTE[1SPC]PROGRAMA[1SPC
]MIDE[1SPC]SU"
40 PRINT"[5SPC]VELOCIDAD[1SPC]ME
CANOGRAFICA"
50 PRINT
90 PRINT:
100 PRINT"[4SPC]SI[1SPC]USTED[1S
PC]QUIERE[1SPC]MEDIR[1SPC]SU[1SP
C]VELOCIDAD"
110 PRINT"[13SPC]PULSE[2SPC]'R'
"
130 PRINT
140 PRINT"[4SPC]PULSANDO[1SPC]OT
RA[1SPC]TECLA[1SPC]PODRA[1SPC]EL
EJIR[1SPC]SU[9SPC]GRADO[1SPC]DE[
1SPC]DIFICULTAD[2SPC]"
180 GET A$:IF A$=""THEN 180
190 IF A$="R" THEN PRINT"R";:GOT
O 900
200 IF ASC(A$)<49ORASC(A$)>57THE
N180
210 RA=10*VAL(A$)
220 PRINTA$;
230 GETA$:IFA$=""THEN230
240 IF ASC(A$)<48 OR ASC(A$)>57
THEN 230
250 PRINTA$;:RA=RA+VAL(A$)
260 GOSUB2000
270 GOSUB3000
280 GOSUB500:
290 N=INT(2000/RA):GOSUB600
300 Y=(Z/LEN(X$))/60
310 RB=10+90/(1+Y)
320 RA=INT(0.8*RA+0.2*RB )
325 D=D+1
330 GOTO260
500 PRINT"[1HOME][3CRSRD]"
505 PRINT"[1SPC]COPIE[1SPC]LAS[1
SPC]SIGUIENTES":PRINT"[1SPC]FRAS
ES[1SPC]":PRINT"[2SPC]PARA[1SPC]
EMPEZAR[1SPC]";
508 PRINT"[2SPC]PULSE[1SPC]ESPAC
IO[1SPC]":PRINT:PRINT
510 R=LEN(X$):J=1
520 IFR<41THEN550
530 PRINTMID$(X$,J,40);
540 J=J+40:R=R-40:GOTO520
550 PRINTMID$(X$,J,R);"[64SPC]"
560 RETURN
565 PRINT"[57SPC]"
600 PRINT"[1HOME][14CRSRD]"
610 POKE1384,160
620 R=LEN(X$)
630 GETA$:IFA$=""THEN630
640 IFA$<>"[1SPC]"THEN630
650 POKE1384,32
660 Z=TI
670 PRINTA$;
680 FORJ=2TOR:W=1383+J
690 POKEW,PEEK(W)+128:T=0
710 GETA$:IFA$<>""THEN740
720 T=T+1:IFT<>NTHEN710
730 Q=245:GOSUB1000:T=0:GOTO710
740 IFA$=MID$(X$,J,1)THEN760
750 Q=210:FORK=1TO5:GOSUB1000:NE
XT:GOTO710
760 PRINTA$;:POKEW,PEEK(W)-128
780 NEXTJ
790 Z=TI-Z:RETURN
800 RA=10:D=-5:GOTO260
900 PRINT"[1CLR][2CRSRD][5SPC]EL
[1SPC]ORDENADOR[1SPC]LE[1SPC]DAR
A[3SPC]"
905 PRINT"[3SPC]ALGUNAS[1SPC]FRA
SES[1SPC]PARA[1SPC]COPIAR[1SPC]T
AN"
907 PRINT"[3SPC]RAPIDO[1SPC]COMO
[1SPC]PUEDA."
910 PRINT:PRINT"[5SPC]EL[1SPC]CU
RSOR[1SPC]BLANCO[1SPC]INDICA[1SP
C]LA[1SPC]PROXIMA"
920 PRINT"[3SPC]LETRA[1SPC](O[1S
PC]ESPACIO)[1SPC]A[1SPC]MECANOGR
AFIAR."
930 PRINT:PRINT"[3SPC]PARA[1SPC]
EMPEZAR[1SPC]PULSE[1SPC]ESPACIO"

940 PRINT:PRINT"[3SPC]AHORA[1SPC
]PULSE[1SPC]CUALQUIER[1SPC]TECLA
[1SPC]PARA[13SPC]EMPEZAR[1SPC]EL
[1SPC]TEST"
946 GETA$:IFA$=""THEN946
950 GOTO800
1000 IF D<0THEN RETURN
1001 POKEV+4,0:POKEV+1,80:POKEV+
4,17
1020 FORT=1TO30:NEXTT
1030 POKEV+1,0:POKEV+4,0
1040 RETURN
1900 REM SELECION PALABRAS ENTRE
 A Y B
1910 RESTORE,
1920 C=A:Q=245
1930 IF C=0 THEN 1950
1940 READ Y$:C=C-1:GOTO1930
1950 C=INT((B-A+1)*RND(1))
1960 IFC=0THEN1980
1970 READY$:C=C-1:Q=230:GOTO1960
```



Dream C: commodore

```
1980 X$=X$+"[1SPC]"+Y$
1990 RETURN
2000 IFD<0THENPRINT"[1CLR][1CRSR
D][1CRSRR]FRASE[3SPC]TEST[1SPC]"
;D+6:RETURN
2010 PRINT"[1CLR][1WHT][9SPC][1C
OMMA][6SHIF*][1COMMS]"
2011 PRINT"VEL.[5SPC][1SHIF-][1S
PC][1YEL][1SPC]";RA;"[1WHT][1SHI
F-]"
2012 PRINT"[9SPC][1COMMZ][6SHIF*
][1COMMX]";"[1COMM8]"
2013 RETURN
3000 X$="":GOSUB4000
3020 IF RND(1)>0.5THEN 3060
3040 GOSUB4200:GOSUB4600
3050 GOTO3070
3060 GOSUB4400
3070 IFRND(1)>0.25THEN3090
3080 GOSUB4800
3090 RETURN
4000 IF RND(1)>0.3THEN4030
4020 GOSUB5000:RETURN
4030 X$=X$+"[1SPC]LA"
4040 IF RND(1)>0.3THEN4060
4050 GOSUB5200
4060 GOSUB5400:RETURN
4200 A=43:B=50:GOSUB1900:RETURN
4400 A=26:B=35:GOSUB1900:RETURN
4600 IF RND(1)>0.6THEN4700
4610 X$=X$+"[1SPC]"+"EL"
4620 IFREN(1)<0.35THENGOSUB5600
4630 GOSUB6000:RETURN
4700 IF RND(1)>0.5THENX$=X$+"[1S
PC]"+"ALGUN":GOTO4720
4710 GOSUB5800
4720 IFRND(1)<0.35THENGOSUB5600
4730 GOSUB6200:RETURN
4800 IF RND(1)>0.7THENX$=X$+"[1S
PC]"+"MUY"
4820 GOSUB 6400:RETURN
5000 A=1:B=10:GOSUB1900:RETURN
5200 A=11:B=17:GOSUB1900:RETURN
5400 A=18:B=25:GOSUB1900:RETURN
5600 A=75:B=79:GOSUB1900:RETURN
5800 A=58:B=68:GOSUB1900:RETURN
6000 A=51:B=57:GOSUB1900:RETURN
6200 A=69:B=74:GOSUB1900:RETURN
6400 A=36:B=42:GOSUB1900:RETURN
9000 DATA ANDRES,JUAN,LUIS,SUSAN
A,JAVIER,VANESSA,GUILLERMO,LUCAS
,ELENA,JOSE
9010 DATALISTO,GORDO,FLACO,ALTO,
BAJO,IGNORANTE,ESTUPIDO
9020 DATACHICO,CHICA,TIO,ESCUELA
,PROFESOR,MAESTRO,REINA,BAILARIN
A
9030 DATAANDUVO,FUE,"CALLE[1SPC]
ABAJO",VOLO,ESCAPO,MARCHO,VIO,HA
BLO,PARO,CANTO
9040 DATARAPIDAMENTE,DESPACIO,"C
ON[1SPC]CUIDADO",DURAMENTE,SUAVE
MENTE,SIN,LIMPIAMENTE
9050 DATADOBLADO,PINCHADO,VOTADO
,VENDIDO,APILADO,PISADO,COLORADO
,DOBLADO
9060 DATAPATO,VINO,CAFE,CORDERO,
CONEJO,CREMA
9070 DATADOS,TRES,4,5,6,7,8,10,1
3,18,20
9080 DATACOCOS,PATATAS,MANZANAS,
FRESAS,RABIOLIS,PASTEL
9090 DATALLENO,VACIO,ROTO,MALO,B
UENO,MEJOR
```

PROGRAMA: UNI3CUEST

```
2 REM UNIT3QUIZ64
4 REM COPYRIGHT (C) ANDREW COLIN
 1982
5 POKE 53281,0
6 POKE 53280,0
10 PRINT"[1CLR][2CRSRD][1WHT][3S
PC]PROGRAMA[1SPC]DE[1SPC]PREGUNT
AS[1SPC]Y[1SPC]RESPUESTAS"
20 PRINT:PRINT"[6SPC]CADA[1SPC]V
EZ,PULSE[1SPC]UNA[1SPC]TECLA[1SP
C]PARA"
30 PRINT"[3SPC]PASAR[1SPC]A[1SPC
]LA[1SPC]PROXIMA[1SPC]PREGUNTA"
40 PRINT:PRINT"[6SPC]SIEMPRE,SUS
[1SPC]RESPUESTAS[1SPC]DEBEN[1SPC
]IR[1SPC]"
50 PRINT"[3SPC]SEGUIDAS[1SPC]DEL
[1SPC][1YEL]RETURN[1COMM8]"
60 PRINT:PRINT:PRINT"[3SPC]BUENA
[1SPC]SUERTE!"
70 FORJ=1TO12:READY$:NEXT
80 READN
90 DIMZ(N)
100 FORJ=1TON:Z(N)=0:NEXT
110 SC=0
120 FORJ=1TON
130 IFZ(J)>=0THEN 200
140 NEXTJ
150 PRINT"[1WHT][1CLR][1CRSRD][7
SPC]SU[1SPC]PUNTUACION";SC
160 PRINT:PRINT"[4SPC]RESPUESTAS
[1SPC]CORRECTAS"
170 PRINT:PRINT"[4SPC]DE[1SPC]UN
[1SPC]TOTAL[1SPC]DE";N
180 PRINT:PRINT"[4SPC]ESTO[1SPC]
ES";INT(100*SC/N+0.5);"POR[1SPC]
CIENTO"
190 END
200 D=1+INT(N*RND(1))
210 IF Z(D)=-1THEN200
220 C=Z(D)+1
230 GOSUB1200
240 FORT=1TO3
250 GETA$:IFA$=""THEN250
260 GOSUB1000
270 GOSUB1300
280 GOSUB1600
285 IFR=1ANDT=1ANDZ(D)=0THEN SC=
SC+1
290 IF R=1 AND T=1 THENZ(D)=-1:G
OTO400
300 IF R=1 THEN 320
310 NEXTT
320 Z(D)=Z(D)+1:GOTO120
400 GOTO120
1000 PRINT"[1CLR]"
1010 PRINT"[1CLR]"
1020 FORJ=0TO39:POKE1024+J,160:P
OKE55296+J,C:NEXT
1030 FORJ=0TO920STEP40:POKE1103+
J,160:POKE55375+J,C:NEXT
1040 FORJ=39TO0STEP-1:POKE1984+J
,160:POKE56256+J,C:NEXT
1050 FORJ=920TO0STEP-40:POKE1064
+J,160:POKE55336+J,C:NEXT
1055 W=2:IFLEN(X$)>120THENW=1
1060 P=1:J=3: K=3
1070 A$=MID$(X$,P,1):P=P+1
1080 IFA$="#"THEN 1150
1090 IFA$="%"THENJ=J+W:K=3:GOTO1
070
1100 POKE1024+40*J+K,ASC(A$)AND
63:POKE55296+40*J+K,15:K=K+1
1110 GOTO1070
1150 RETURN
1200 RESTORE
1215 FORJ=1TO12:READX$:NEXT
1220 READN
1230 FORJ=1TOD
1240 READX$:READY$
1245 X$=X$+Y$:READY$
1250 NEXT J
1260 RETURN
1300 PRINT"[1HOME][16CRSRD][3CRS
RR][1COMM+][1CRSRL]";
1305 A$=""
1307 GETQ$:IFQ$=""THEN 1307
1309 IFASC(Q$)<>20 THEN1314
1310 IFA$=""THEN1307
1311 PRINT"[1CRSRL][1SPC][1CRSRL
]";:A$=LEFT$(A$,LEN(A$)-1):GOTO1
307
1314 IFASC(Q$)=13 THEN 1321
1315 IF Q$<"0" OR (Q$>"9" AND Q$
<"A")OR Q$>"Z" THEN 1307
1316 PRINTQ$;
1317 A$=A$+Q$:GOTO1307
1321 P=LEN(A$):IF P=0THEN1300
1322 IFASC(RIGHT$(A$,1))<>32THEN
1330
1323 A$=MID$(A$,1,P-1)
1326 GOTO1321
1330 IF LEFT$(Y$,1)<>"%"THEN1380

1340 P=2
1350 IFMID$(Y$,P,1)<>"-"THENP=P+
1:GOTO1350
1360 IFVAL(A$)>=VAL(MID$(Y$,2,P-
1))ANDVAL(A$)<=VAL(MID$(Y$,P+1,L
EN(Y$))) THEN1500
1370 R=0:RETURN
1380 P=1
1390 IFMID$(Y$,P,1)="#"THEN1370
1400 Q=P
1410 IFMID$(Y$,Q,1)<>"%"THENQ=Q+
```

```
1:GOTO1410
1420 IFMID$(Y$,P,Q-P)=A$THEN1500

1430 P=Q+1:GOTO1390
1500 :R=1:RETURN
1600 RESTORE
1620 S=6*(1-R)+INT(6*RND(1))+1
1630 FORJ=1TOS:READZ$:NEXT
1640 PRINT"[1COMM8][1HOME][17CRS
RD][2CRSRR]";Z$
1650 IFT<=2 THEN GOTO1750
1660 PRINT"[2CRSRR]LA[1SPC]RESPU
ESTA[1SPC]CORRECTA[1SPC]ES":PRIN
T"[2CRSRR]";
1670 IF LEFT$(Y$,1)="%"THEN1710
1680 P=1
1690 IFMID$(Y$,P,1)="%"THENGOTO1
750
1700 PRINTMID$(Y$,P,1);:P=P+1:GO
TO1690
1710 PRINTMID$(Y$,2,LEN(Y$)):GOT
O1750
1750 PRINT"[1HOME][21CRSRD][1CRS
RR][1YEL][3SPC]AHORA[1SPC]PULSE[
1SPC]UNA[1SPC]TECLA":RETURN
2000 X$="ABCD"
2010 INPUTY$
2020 IFX$=Y$THEN2050
2030 PRINT"NO":GOTO2000
2050 PRINT"SI"
2060 STOP
8010 DATASI,CORRECTO,OK,BIEN,"AS
I[1SPC]ES","VALIDA[1SPC]LA[1SPC]
RESPUESTA"
8020 DATANO,MAL,"PESIMO",INCORRE
CTO,"NO[1SPC]ES[1SPC]ASI","NO[1S
PC]ES[1SPC]VALIDA[1SPC]LA[1SPC]R
ESPUESTA"
9000 DATA13
9010 DATA"SI[1SPC]EL[1SPC]CBM[1S
PC]64[1SPC]SE[1SPC]BLOQUEA[1SPC]
%QUE[1SPC]DEBE[1SPC]HACER[1SPC]P
ARA[1SPC]LIBERARLO%A)PULSAR[1SPC
]RETURN
9015 DATA"%B)APAGARLO[1SPC](OFF)
%C)PULSAR[1SPC]RUN/STOP%#",B%#
9020 DATA"CUAL[1SPC]ES[1SPC]EL[1
SPC]ANCHO[1SPC]DE[1SPC]PANTALLA,
[1SPC]EN%CARACTERES[1SPC]?",%#,4
0%#
9030 DATA"CUANTAS[1SPC]LINEAS[1S
PC]DE[1SPC]CARACTERES%TIENE[1SPC
]LA[1SPC]PANTALLA?",%#,25%#
9040 DATA"PULSARIA[1SPC]LA[1SPC]
TECLA[1SPC]SHIFT[1SPC]PARA%MOVER
[1SPC]EL[1SPC]CURSOR[1SPC]A[1SPC
]LA[1SPC]IZQUIERDA",%#,SI%S%#
9050 DATA"PULSARIA[1SPC]LA[1SPC]
TECLA[1SPC]SHIFT[1SPC]PARA%MOVER
[1SPC]EL[1SPC]CURSOR[1SPC]A[1SPC
]LA[1SPC]DERECHA",%#,NO%N%#
9060 DATA"PULSARIA[1SPC]LA[1SPC]
TECLA[1SPC]SHIFT[1SPC]PARA%MOVER
[1SPC]EL[1SPC]CURSOR[1SPC]HACIA[
1SPC]ARRIBA",%#,SI%S%#
9070 DATA"PULSAR[A[1SPC]LA[1SPC]
TECLA[1SPC]SHIFT[1SPC]PARA%MOVER
[1SPC]EL[1SPC]CURSOR[1SPC]HACIA[
1SPC]ABAJO[1SPC]",%#,NO%N%#
9080 DATA"QUE[1SPC]PASA[1SPC]AL[
1SPC]PULSAR[1SPC]LA[1SPC]TECLA[1
SPC]CLR/HOME%SOLA?%A)LIMPIA[1SPC
]PANTALLA[1SPC]%B)COLOCA[1SPC]EL

9085 DATA"[1SPC]CURSOR[1SPC]EN[1
SPC]LA[1SPC]ESQUINA%SUPERIOR[1SP
C]IZQUIERDA%C)BORRA[1SPC]LA[1SPC
]LINEA[1SPC]%#",B%#
9090 DATA"COMO[1SPC]LIMPIARIA[1S
PC]LA[1SPC]PANTALLA?%%A)APAGANDO
%%B)PULSANDO[1SPC]SHIF[1SPC]Y[1S
PC]"
9095 DATA"INST/DEL%%C)[1SPC]CON[
1SPC]SHIFT[2SPC]Y[1SPC]CLR/HOME%
#",C%#
9100 DATA"CUANTOS[1SPC]COLORES[1
SPC]TIENE%EL[1SPC]COMMODORE[1SPC
]64%A)[1SPC]32[10SPC]%B)[1SPC]16
[3SPC]%C
9105 DATA")[1SPC]8?%#","B%16%#"
9110 DATA"QUE[1SPC]COMANDO[1SPC]
PONE[1SPC]EL[1SPC]FONDO[1SPC]BLA
NCO?%A)[1SPC]POKE[1SPC]53280,2
9112 DATA"%B)[1SPC]POKE[1SPC]523
80,2%C)[1SPC]POKE[1SPC]53280,1%D
)[1SPC]POKE[1SPC]53281,1%#",C%#
9120 DATA"COMO[1SPC]PONE[1SPC]EL
[1SPC]CURSOR[1SPC]ROJO[1SPC]?%A)
CONTROL[1SPC]Y[1SPC]RED%B)CONTRO
L[1SPC]Y[1SPC]2%C)SHIFT[1SPC]Y
9125 DATA"[1SPC]RED%D)C=[1SPC]Y[
1SPC]RED%#","A%#"
9130 DATA"QUE[1SPC]HACE[1SPC]EST
E[1SPC]COMANDO[1SPC]?%%[5SPC]POK
E[1SPC]53281,8%%A)[1SPC]FONDO[1S
PC]VERDE
9135 DATA"%B)[1SPC]BORDE[1SPC]NA
RANJA%C)[1SPC]FONDO[1SPC]NARANJA
[7SPC]%#",C%#
```

PROGRAMA: UNI4EJER

```
0 REM UNI4EJER 64
1 REM COPYRIGHT (C) ANDREW COLIN
 1982
3 POKE 53281,0
4 POKE 53280,0
5 PRINT"[1CLR][1WHT]POR[1SPC]FAV
OR[1SPC]ENTRE[1SPC]UN":PRINT"NIV
EL[1SPC]DE[1SPC]DIFICULTA"
6 PRINT"(1[1SPC]SI[1SPC]ESTA[1SP
C]PROVANDO":PRINT"LA[1SPC]PRIMER
A[1SPC]VEZ":INPUTDM
7 IFDM>0AND DM<9 THEN 9
8 PRINT"NO[1SPC]VALIDO.OTRA[1SPC
]VEZ":INPUTDM:GOTO7
9 DIMS%(15),S$(15)
10 FORBB=1TO20
11 U$=CHR$(34):D$=CHR$(32)
12 P=1:D=1:IFDM<1 THEN DM=1
13 PRINT"[1CLR]";MID$("[1WHT][1R
ED][1CYN][1PUR][1GRN][1BLU][1YEL
][1COMM1][1COMM2][1COMM3][1COMM4
][1COMM5][1COMM6][1COMM7][1COMM8
]",DM,1);"[8CRSRR][1RVSON][24SPC
]"
14 FORJ=1TO21:PRINT"[8CRSRR][1RV
SON][1SPC][1RVSOF][22SPC][1RVSON
][1SPC]"
15 NEXTJ
16 PRINT"[8CRSRR][1RVSON][24SPC]
[1WHT]"
19 U$=CHR$(34):D$=CHR$(32)
20 P=1:D=1
22 PRINT"[1HOME][1CRSRD][9CRSRR]
NIVEL[1SPC]DE[1SPC]DIFICULTAD":P
RINT"[11CRSRR][1SPC]=[1SPC]";DM
23 PRINT"[1CRSRD][9CRSRR][1SPC]E
SCRIBA[1SPC]EXACTAMENTE":PRINT"[
9CRSRR][1SPC]EL[1SPC]SIGUIENTE"
25 PRINT"[9CRSRR][2SPC]COMANDO[1
SPC]QUE[4SPC]"
26 PRINT"[9CRSRR][1SPC]DARA[1SPC
]:[1SPC]-":PRINT"[9CRSRR][1SPC]S
EGUIDO[1SPC]DE[3SPC]"
27 PRINT"[9CRSRR][1SPC]RETURN":P
RINT
30 FORJ=1TO10:N%(J)=0:N$(J)="":M
(J)=0:M$(J)="":MS$="":NEXT
35 Q$="+"+U$+"[1SPC]"+U$+"+"
36 A$="+"+U$+"&"+U$+"+"
40 P$="PRINT[1SPC]":W$=""
50 IF RND(1)<0.5 THEN 70
60 GOSUB1000
65 P$=P$+X$:W$=W$+STR$(X)+D$:IFX
<-100ORX>100THEN20
66 GOTO80
70 GOSUB2000:P$=P$+X$:W$=W$+Y$
80 IF RND(1)>.5ORD>=DMTHEN130
90 P$=P$+";"
100 IFRND(1)<.5THEN120
105 GOSUB1000
110 P$=P$+X$:W$=W$+STR$(X):IFX<-
100ORX>100THEN20
115 GOTO130
120 GOSUB2000:P$=P$+X$:W$=W$+Y$
130 REM
140 FORE=20TO1STEP-1
150 FORJ=1TO10
160 IFN%(J)=ETHENPRINT"[9CRSRR]"
;N$(J)
170 IFM(J)=ETHEN PRINT"[9CRSRR]"
;M$(J)
180 NEXTJ,E
190 PRINT:PRINT"[9CRSRR]";P$
200 G$="":PRINT"[9CRSRR]";
201 GETH$:A=RND(1):IFH$=""THEN20
1
202 IFASC(H$)=13THEN210
203 IFASC(H$)<>20THEN 206
204 IFLEN(G$)=0THEN201
205 G$=MID$(G$,1,LEN(G$)-1):PRIN
T"[1CRSRL][1SPC][1CRSRL]";:GOTO2
01
206 PRINT H$;:G$=G$+H$:GOTO201
210 J=LEN(G$):IFRIGHT$(G$,1)=D$T
HENG$=MID$(G$,1,J-1):GOTO210
220 K=LEN(W$):IFRIGHT$(W$,1)=D$T
HENW$=MID$(W$,1,K-1):GOTO220
230 IFG$=W$THEN400
240 GG$=""
250 FORL=1TOJ
260 IFMID$(G$,L,1)<>D$THENGG$=GG
$+MID$(G$,L,1)
270 NEXT L
280 WW$=""
290 FORL=1TOK
300 IFMID$(W$,L,1)<>D$THENWW$=WW
$+MID$(W$,L,1)
310 NEXTL
320 IFGG$=WW$THEN 600
330 PRINT:PRINT"[9CRSRR]";"MAL.[
1SPC]LA[1SPC]RESPUESTA[1SPC]ES"
340 PRINT"[9CRSRR]";W$
345 IF DM>0.25THENDM=DM-0.25
350 PRINT:PRINT"[9CRSRR]";"PULSE
[1SPC]UNA[1SPC]TECLA"
360 GETA$:IFA$=""THEN360
370 GOTO900
400 PRINT:PRINT"[9CRSRR]";"CORRE
CTO"
410 IFDM<9THENDM=DM+0.25
420 GOTO350
600 PRINT:PRINT"[9CRSRR]CORRECTO
[1SPC]SALVO[1SPC]POR":PRINT"[9CR
SRR]EL[1SPC]ESPACIO[1SPC]DERECHO
"
```

```
605 PRINT"[9CRSRR]LA[1SPC]RESPUE
STA[1SPC]ES"
610 PRINT:PRINT"[9CRSRR]";W$
620 GOTO350
900 NEXTBB
910 PRINT"[1CLR][3CRSRD][9CRSRR]
EN[1SPC]20[1SPC]PREGUNTAS[1SPC]S
U":PRINT"[9CRSRR]NIVEL[1SPC]DE[1
SPC]DIFICULTAD"
915 PRINT"[9CRSRR]HA[1SPC]VARIAD
O[1SPC]A";DM
920 PRINT"[9CRSRR]ENTRE[1SPC]LA[
1SPC]PROXIMA[1SPC]VEZ":PRINT"[9C
RSRR]ESTE[1SPC]NIVEL"
930 END
1000 IFRND(1)<.7ANDD<DMTHEND=D+1
:GOTO1030
1010 IF RND(1)<0.7 AND D<DM THEN
 D=D+1:GOTO1030
1020 GOSUB1500:RETURN
1030 GOSUB1500:XX$=X$:XX=X
1040 GOSUB1500
1050 IFRND(1)>0.6 THEN 1070
1060 X$=XX$+"+"+X$:X=XX+X:D=D-1:
RETURN
1070 IFRND(1)<0.5THEN1090
1080 X$=XX$+"-"+X$:X=XX-X:RETURN

1090 X$=XX$+"*"+X$:X=XX*X:D=D-1:
RETURN
1500 IFRND(1)<.5ANDD<DMTHEND=D+1
:GOTO1600
1510 IFRND(1)<.5ANDD<DMTHEND=D+1
:GOTO1600
1520 X=INT(10*RND(1))
1530 X$=CHR$(X+48):RETURN
1600 V=1+INT(10*RND(1))
1610 V=1+INT(10*RND(1))
1620 IFN%(V)>0ANDN$(V)=""THEN161
0
1630 IFN%(V)=0THEN1650
1635 IFN%(V)<DTHEN1610
1640 X$=CHR$(V+64):X=NS%(V):D=D-
1:RETURN
1650 N%(V)=D
1660 S%(P)=V:S%(P+1)=XX:S$(P)=XX
$:P=P+2
1670 GOSUB1000
1680 P=P-2:V=S%(P):XX=S%(P+1):XX
$=S$(P)
1690 NS%(V)=X:N$(V)=CHR$(V+64)+"
[1SPC]=[1SPC]"+X$
1700 X$=CHR$(V+64):D=D-1:RETURN
2000 IFRND(1)<.5ANDD<DMTHEND=D+2
:GOTO2030
2010 IFRND(1)<0.5AND D<DM THEND=
D+2:GOTO2030
2020 GOSUB2500:RETURN
2030 GOSUB2500:XX$=X$:YY$=Y$
2040 GOSUB2500:IFRND(1)<0.5THEN2
060
2050 X$=XX$+Q$+X$:Y$=YY$+"[1SPC]
"+Y$:RETURN
2060 X$=XX$+A$+X$:Y$=YY$+"&"+Y$:
RETURN
2500 IFRND(1)<.5ANDD<DMTHEND=D+2
:GOTO2590
2510 IFRND(1)<0.5 ANDD<DMTHEND=D
+2:GOTO2590
2520 Y$=CHR$(26*RND(1)+65)
2580 X$=CHR$(34)+Y$+CHR$(34):RET
URN
2590 V=1+INT(10*RND(1))
2600 IFM(V)>0ANDM$(V)=""THEN 259
0
2610 IFM(V)=0THEN2630
2620 X$=CHR$(V+64)+"$":Y$=MS$(V)
:RETURN
2630 M(V)=D:S%(P)=V:S$(P)=XX$:S$
(P+1)=YY$:P=P+2
2640 GOSUB2000
2650 P=P-2:V=S%(P):XX$=S$(P):YY$
=S$(P+1)
2660 MS$(V)=Y$:M$(V)=CHR$(V+64)+
"$="+X$
2670 X$=CHR$(V+64)+"$":RETURN
```

PROGRAMA: UNISCUEST

```
1 RESTORE
2 REM UNISCUEST 64
3 REM COPYRIGHT (C) ANDREW COLIN
 1982
5 POKE 53281,0
6 POKE 53280,0
10 PRINT"[1CLR][1CRSRD][1WHT][3S
PC]PROGRAMA[1SPC]DE[1SPC]PREGUNT
AS[1SPC]Y[1SPC]RESPUESTAS."
20 PRINT:PRINT"[5SPC]CADA[1SPC]V
EZ,[1SPC]PULSE[1SPC]UNA[1SPC]TEC
LA[1SPC]PARA"
30 PRINT"[2SPC]TOMAR[1SPC]LA[1SP
C]SIGUIENTE[1SPC]PREGUNTA"
40 PRINT:PRINT"[6SPC]SIEMPRE[1SP
C]SIGA[1SPC]SU[1SPC]RESPUESTA"
50 PRINT"[3SPC]DE[1SPC]LA[1SPC]T
ECLA[1SPC][1YEL]RETURN[1COMM8][1
SPC]"
60 PRINT:PRINT:PRINT"[3SPC]BUENA
[1SPC]SUERTE"
70 FORJ=1TO12:READY$:NEXT
80 READN
90 DIMZ(N)
100 FORJ=1TON:Z(N)=0:NEXT
110 SC=0
120 FORJ=1TON
130 IFZ(J)>=0THEN 200
140 NEXTJ
150 PRINT"[1WHT][1CLR][1CRSRD][9
SPC]SU[1SPC]PUNTUACION";SC
160 PRINT:PRINT"[4SPC]RESPUESTAS
[1SPC]CORRECTAS[1SPC]"
170 PRINT:PRINT"[4SPC]A[1SPC]LA[
1SPC]PRIMERA";N
180 PRINT:PRINT"[4SPC]ESTO[1SPC]
ES";INT(100*SC/N+0.5);"%[6SPC]"
190 END
200 D=1+INT(N*RND(1))
210 IF Z(D)=-1THEN200
220 C=Z(D)+1
230 GOSUB1200
240 FORT=1TO3
250 GETA$:IFA$=""THEN250
260 GOSUB1000
270 GOSUB1300
280 GOSUB1600
285 IFR=1ANDT=1ANDZ(D)=0THEN SC=
SC+1
290 IF R=1 AND T=1 THENZ(D)=-1:G
OTO400
300 IF R=1 THEN 320
310 NEXTT
320 Z(D)=Z(D)+1:GOTO120
400 GOTO120
1000 PRINT"[1CLR]"
1010 PRINT"[1CLR]"
1020 FORJ=0TO39:POKE1024+J,160:P
OKE55296+J,C:NEXT
1030 FORJ=0TO920STEP40:POKE1103+
J,160:POKE55375+J,C:NEXT
1040 FORJ=39TO0STEP-1:POKE1984+J
,160:POKE56256+J,C:NEXT
1050 FORJ=920TO0STEP-40:POKE1064
+J,160:POKE55336+J,C:NEXT
1055 W=2:IFLEN(X$)>120THENW=1
1060 P=1:J=3: K=3
1070 A$=MID$(X$,P,1):P=P+1
1080 IFA$="#"THEN 1150
1090 IFA$="%"THENJ=J+W:K=3:GOTO1
070
1100 POKE1024+40*J+K,ASC(A$)AND
63:POKE55296+40*J+K,15:K=K+1
1110 GOTO1070
1150 RETURN
1200 RESTORE
1215 FORJ=1TO12:READX$:NEXT
1220 READN
1230 FORJ=1TOD
1240 READX$:READY$
1245 X$=X$+Y$:READY$
1250 NEXT J
1260 RETURN
1300 PRINT"[1HOME][16CRSRD][3CRS
RR][1COMM+][1CRSRL]";
1305 A$=""
1307 GETQ$:IFQ$=""THEN 1307
1309 IFASC(Q$)<>20 THEN1314
1310 IFA$=""THEN1307
1311 PRINT"[1CRSRL][1SPC][1CRSRL
]";:A$=LEFT$(A$,LEN(A$)-1):GOTO1
307
1314 IFASC(Q$)=13 THEN 1321
1315 IF Q$<"0" OR (Q$)"9" AND Q$
<"A")OR Q$>"Z" THEN 1307
1316 PRINTQ$;
1317 A$=A$+Q$:GOTO1307
1321 P=LEN(A$):IF P=0THEN1300
1322 IFASC(RIGHT$(A$,1))<>32THEN
1330
1323 A$=MID$(A$,1,P-1)
1326 GOTO1321
1330 IF LEFT$(Y$,1)<>"%"THEN1380

1340 P=2
1350 IFMID$(Y$,P,1)<>"-"THENP=P+
1:GOTO1350
1360 IFVAL(A$)>=VAL(MID$(Y$,2,P-
1))ANDVAL(A$)<=VAL(MID$(Y$,P+1,L
EN(Y$))) THEN1500
1370 R=0:RETURN
1380 P=1
1390 IFMID$(Y$,P,1)="#"THEN1370
1400 Q=P
1410 IFMID$(Y$,Q,1)<>"%"THENQ=Q+
```

```
1:GOTO1410
1420 IFMID$(Y$,P,Q-P)=A$THEN1500

1430 P=Q+1:GOTO1390
1500 :R=1:RETURN
1600 RESTORE
1620 S=6*(1-R)+INT(6*RND(1))+1
1630 FORJ=1TOS:READZ$:NEXT
1640 PRINT"[1COMM8][1HOME][17CRS
RD][2CRSRR]";Z$
1650 IFT<=2 THEN GOTO1750
1660 PRINT"[2CRSRR]LA[1SPC]RESPU
ESTA[1SPC]CORRECTA[1SPC]ES":PRIN
T"[2CRSRR]";
1670 IF LEFT$(Y$,1)="%"THEN1710
1680 P=1
1690 IFMID$(Y$,P,1)="%"THENGOTO1
750
1700 PRINTMID$(Y$,P,1);:P=P+1:GO
TO1690
1710 PRINTMID$(Y$,2,LEN(Y$)):GOT
O1750
1750 PRINT"[1HOME][21CRSRD][1CRS
RR][1YEL][2SPC]AHORA[1SPC]PULSE[
1SPC]UNA[1SPC]TECLA[1SPC]":RETUR
N
2000 X$="ABCD"
2010 INPUTY$
2020 IFX$=Y$THEN2050
2030 PRINT"NO":GOTO2000
2050 PRINT"YES"
2060 STOP
8010 DATASI,CORRECTO,BIEN,FABULO
SO,"ES[1SPC]ASI",OK
8020 DATANO,MAL,"NO[1SPC]ES[1SPC
]ASI",FATAL,"MUY[1SPC]MAL",INCOR
RECTO
9000 DATA9
9010 DATA"[2SPC]ES%[7SPC]X+1=X%[
1SPC]UN[1SPC]COMANDO[1SPC]LEGAL[
1SPC]BASIC?","%[1SPC]A)[1SPC]SI%
[1SPC]B)[1SPC]NO%#"
9015 DATA"B%N%NO%#"
9020 DATA"[2SPC]QUE[1SPC]RAPIDEZ
[1SPC]TIENE[1SPC]EL[1SPC]CBM[1SP
C]64%%(COMMANDOS[1SPC]POR[1SPC]S
EGUNDO)%%[1SPC]CANTIDAD"
9025 DATA"[1SPC]APROXIMADA%#","%
50-150#"
9030 DATA"[2SPC]CUANTOS[1SPC]COM
ANDOS[1SPC]PUEDE[1SPC]ALMACENAR%
[1SPC]LA[1SPC]MAQUINA[1SPC]A[1SP
C]LA[1SPC]VEZ[1SPC]?%%A)[1SPC]TA
NTOS
9035 DATA"[1SPC]COMO[1SPC]NECESI
TE[1SPC]%%B)[1SPC]10[1SPC]%%C)TA
NTOS[1SPC]COMO[1SPC]SE[1SPC]PUED
AN[1SPC]ALMACENAR%%#","C%#"
9040 DATA"LOS[1SPC]COMANDOS[1SPC
]SON[1SPC]ALMACENADOS%A)[1SPC]EN
[1SPC]EL[1SPC]ORDEN[1SPC]ESCRITO
[1SPC]%B)EN[1SPC]SECUENCIA[1SPC]
DESCE
9045 DATA"NDENTE%[1SPC]DE[1SPC]V
ARIABLES[1SPC]NUMERICAS%C)[1SPC]
NINGUNA[1SPC]DE[1SPC]ESTAS%#","C
%#"
9050 DATA"[2SPC]CUALQUIER[1SPC]V
ARIABLE[1SPC]NUMERICA[1SPC]%DEBE
[1SPC]FINALIZAR%CON[1SPC]UN[1SPC
]CERO%%[1SPC]CIERTO[2SPC]"
9055 DATA"O[1SPC]FALSO%%#","FALS
O%F%#"
9060 DATA"%COMO[1SPC]PUEDE[1SPC]
SU[1SPC]ORDENADOR%[3SPC]IR[1SPC]
MAS[1SPC]LENTO?%%A)[1SPC]CON[1SP
C]LA[1SPC]TECLA[1SPC]INST/DEL%"
9065 DATA"B)[1SPC]CON[1SPC]LA[1S
PC]TECLA[1SPC]CONTROL%C)[1SPC]PU
LSANDO[1SPC]EL[1SPC]ESPACIO[1SPC
]%#","B%CONTROL%#"
9070 DATA"[1SPC]UNA[1SPC]COMA[1S
PC]DESPUES[1SPC]DE[1SPC]UN[1SPC]
STRING%SIGNIFICA%%A)[1SPC]SALTA[
1SPC]LINEA%B)[1SPC]NO[1SPC]SALTA

9075 DATA"[1SPC]A[1SPC]NUEVA[1SP
C]LINEA%C)[1SPC]REEMPEZAR[1SPC]E
STA[1SPC]LINEA%%#","B%#"
9080 DATA"[1SPC]UNA[1SPC]VARIABL
E[1SPC]PUEDE[1SPC]TENER%A)[1SPC]
UN[1SPC]SOLO[1SPC]VALOR[1SPC]%B)
[1SPC]VARIOS[1SPC]VALORES,[1SPC]
DESPUES
9085 DATA"%DE[1SPC]OTROS%C)[1SPC
]VARIOS[1SPC]VALORES[1SPC]A[1SPC
]LA[1SPC]VEZ%#","B%#"
9090 DATA"[3SPC]ES%%[5SPC]LET[1S
PC]G[1SPC]=[1SPC]G[1SPC]+[1SPC]5
%%UN[1SPC]COMANDO[1SPC]BASIC[1SP
C]?%"
9095 DATA"%#","SI%S%CIERTO%#"
```

PROGRAMA: FRASES

```
3 REM FRASES (UNIDAD 6)
5 REM COPYRIGHT (C) ANDREW COLIN
 1981
10 N=13
20 DIMP(13,2)
30 GOSUB2000
40 GOSUB100
45 PRINT"[1CLR]"
46 X$=MID$(X$,2,LEN(X$)-1)+"."
50 GOSUB5000
60 PRINT:PRINT:PRINT:PRINT:PRINT
:PRINT:PRINT"PARA[1SPC]LA[1SPC]P
ROXIMA[1SPC]FRASE"
70 PRINT"PULSE[1SPC]UNA[1SPC]TEC
LA"
80 X=RND(1)
90 GETA$:IFA$=""THEN80
95 GOTO40
100 REM COMO GENERADOR DE FRASES

105 X$=""
110 IF RND(1)<0.5THEN300
120 GOSUB1100
130 IFRND(1)<0.3THEN700
140 IFRND(1)<0.5THEN170
150 A=2:GOSUB1000
160 GOSUB1200:GOTO800
170 A=3:GOSUB1000
180 A=4:GOSUB1000
190 GOTO800
300 GOSUB1200
310 IFRND(1)<0.3THEN700
320 IFRND(1)>0.5THEN370
330 A=2:GOSUB1000
340 GOSUB1100
350 GOTO800
370 A=3:GOSUB1000
380 A=5:GOSUB1000
390 GOTO800
700 A=1:GOSUB1000:GOTO900
800 IFRND(1)<0.6THEN RETURN
810 A=6:GOSUB1000:RETURN
900 IFRND(1)>0.5THEN RETURN
910 A=7:GOSUB1000:RETURN
1000 REM SUBRUTINA PARA SELECCIO
NAR PALABRAS DE CATEGORIA A ALEA
TORIAMENTE
1005 Y=P(A,1)+INT((P(A,2)-P(A,1)
+1)*RND(1))
1010 RESTORE:FORZ=1TOY:READY$:NE
XT
1020 X$=X$+"[1SPC]"+Y$
1030 RETURN
1100 REM SELECCION SUJETO MASCUL
INO
1105 IFRND(1)>0.5THEN1120
1110 A=8:GOSUB1000:RETURN
1120 X$=X$+"[1SPC]EL"
1130 IF RND(1)<0.4THENA=9:GOSUB1
000
1140 A=10:GOSUB1000
1150 RETURN
1200 REM SELECCION SUJETO FEMENI
NO
1205 IFRND(1)>0.5THEN1220
1210 A=11:GOSUB1000:RETURN
1220 X$=X$+"[1SPC]LA"
1230 IFRND(1)<0.4THENA=12:GOSUB1
000
1240 A=13:GOSUB1000:RETURN
2000 REM PONER PUNTERO TABLA
2010 X=1
2020 FORJ=1TON
2030 P(J,1)=X
2040 READX$
2050 IFX$<>"Z"THENX=X+1:GOTO2040

2060 P(J,2)=X-1:X=X+1
2070 NEXTJ
2090 RETURN
5000 REM MOSTRAR FRASE X$ SIN RO
TURA A TRAVES DE LINEAS
5005 IFLEN(X$)>40THEN5020
5010 PRINTX$:RETURN
5020 IF MID$(X$,41,1)<>" "THEN 5
050
5030 PRINTMID$(X$,1,40)
5040 X$=RIGHT$(X$,LEN(X$)-41):PR
INT
5045 GOTO5000
5050 IFMID$(X$,41,1)<>"[1SPC]"TH
EN5080   THEN 5000
5060 PRINTMID$(X$,1,40)
5065 X$=MID$(X$,41,LEN(X$)-40):P
RINT
5070 GOTO5000
5080 P=39
5090 IFMID$(X$,P,1)<>"[1SPC]"THE
NP=P-1:GOTO5090   GOTO 5000
5100 PRINTMID$(X$,1,P-1) :PRINT
5110 X$=MID$(X$,P+1,LEN(X$)-P)
5120 GOTO5000
9000 DATAFUMA,CANTA,ZUMBA,TOCA E
L BANJO,ENCIENDE LA RADIO,SILBA,
Z
9010 DATACOMIO,AGITO LAS MANOS C
ON,BEBIO CON,SALUDO,BESO,ATACO
9011 DATAADMIRABA,DESPRECIABA,NO
 RECONOCIA,ANDABA HACIA,Z
9020 DATATOMO DE, SACO,TIRO,VEND
IO,LAVO,APOSTO,Z
9030 DATASU CAMISA,SUS ZAPATOS,S
U ABRIGO,SUS CALCETINES,SU TUPE,
SU RELOJ,Z
9040 DATASU PELUCA,SU SOMBRERO,S
```



Dream commodore

```
US GAFAS,SUS CALCETINES,SU PERFU
ME,Z
9050 DATADELICIOSO,EN EL PARQUE,
A DIARIO,AYER,LA SEMANA PASADA,H
ACE UNA HORA
9051 DATAEN LA BIBLIOTECA,EN LA
PISCINA,EN LA TIENDA DE ORDENADO
RES,Z
9060 DATATODO EL TIEMPO,A MENUDO
,EN LA DUCHA,LENTAMENTE,SUAVEMEN
TE,Z
9070 DATAPRES. REAGAN,SR.ANDROPO
V,EL AYATOLLAH KHOMEINI,SR. FRAG
A
9071 DATAPOLICIA,CORONEL GHADAFI
,REY,FELIPE GONZALEZ,Z
9080 DATA HABIL,ASPERO,LLENO,GOR
DO,ESTUPIDO,INGENIOSO,ANIMOSO,SU
CIO,ELEGANTE,Z
9090 DATAINGENIERO,PILOTO,PROFES
OR,TEC.LABORATORIO,FUTBOLISTA,LI
BRERO
9091 DATACOMMODORE 64,VENDEDOR,Z

9100 DATASRA.THATCHER,JANE FONDA
,VANESSA REDGRAVE,BO DEREK,
9101 DATAPRINCESA,GLENDA JACKSON
,Z
9110 DATABONITA,PRECIOSA,FEA,FAS
CINANTE,SESUDA,DEMENTE,Z
9120 DATACHICA,MUJER POLICIA,CON
DUCTORA,SECRETARIA,SRTA.,VIUDA,C
HACHA
9130 DATAPELUQUERA,TAXISTA,CAMIO
NERA,FARMACEUTICA,VIUDA,Z
```



PROGRAMA: UNI7CUEST

```
2 REM UNIT7QUIZ64
3 REM COPYRIGHT (C) ANDREW COLIN
 1982
5 POKE 53281,0
6 POKE 53280,0
10 PRINT"[1CLR][1CRSRD][1WHT][3S
PC]PROGRAMA[1SPC]DE[1SPC]PREGUNT
AS[1SPC]Y[1SPC]RESPUESTAS"
20 PRINT:PRINT"[5SPC]CADA[1SPC]V
EZ[1SPC]PULSE[1SPC]UNA[1SPC]TECL
A[1SPC]PARA"
30 PRINT"[3SPC]ACCEDER[1SPC]A[1S
PC]LA[1SPC]PROXIMA[1SPC]PREGUNTA
[1SPC]"
40 PRINT:PRINT"[5SPC]SU[1SPC]RES
PUESTA[1SPC]SIEMPRE[1SPC]SEGUIDA
[20SPC]DE[1SPC]LA[1SPC]TECLA"
50 PRINT"[16SPC][1YEL]RETURN[1CO
MM8][1SPC]"
60 PRINT:PRINT:PRINT"[1SPC]BUENA
[1SPC]SUERTE"
70 FORJ=1TO12:READY$:NEXT
80 READN
90 DIMZ(N)
100 FORJ=1TON:Z(N)=0:NEXT
110 SC=0
120 FORJ=1TON
130 IFZ(J)>=0THEN 200
140 NEXTJ
150 FORJ=1TO2000:NEXTJ
155 PRINT"[1WHT][1CLR][1CRSRD][5
SPC]SU[1SPC]PUNTUACION";SC
160 PRINT:PRINT"[4SPC]CUESTIONES
[1SPC]ACERTADAS[1SPC]"
170 PRINT:PRINT"[3SPC]A[1SPC]LA[
1SPC]PRIMERA";N
180 PRINT:PRINT"[4SPC]ESTO[1SPC]
ES";INT(100*SC/N+0.5);"%"
190 END
200 D=1+INT(N*RND(1))
210 IF Z(D)=-1THEN200
220 C=Z(D)+1
230 GOSUB1200
240 FORT=1TO3
250 GETA$:IFA$=""THEN250
260 GOSUB1000
270 GOSUB1300
280 GOSUB1600
285 IFR=1ANDT=1ANDZ(D)=0THEN SC=
SC+1
290 IF R=1 AND T=1 THENZ(D)=-1:G
OTO400
300 IF R=1 THEN 320
310 NEXTT
320 Z(D)=Z(D)+1:GOTO120
```

```
400 GOTO120
1000 PRINT"[1CLR]"
1010 PRINT"[1CLR]"
1020 FORJ=0TO39:POKE1024+J,160:P
OKE55296+J,C:NEXT
1030 FORJ=0TO920STEP40:POKE1103+
J,160:POKE55375+J,C:NEXT
1040 FORJ=39TO0STEP-1:POKE1984+J
,160:POKE56256+J,C:NEXT
1050 FORJ=920TO0STEP-40:POKE1064
+J,160:POKE55336+J,C:NEXT
1055 W=2:IFLEN(X$)>110THENW=1
1060 P=1:J=3: K=3
1070 A$=MID$(X$,P,1):P=P+1
1080 IFA$="#"THEN 1150
1090 IFA$="%"THENJ=J+W:K=3:GOTO1
070
1095 IFA$="£"THEN A$=CHR$(34)
1100 POKE1024+40*J+K,ASC(A$)AND
63:POKE55296+40*J+K,15:K=K+1
1110 GOTO1070
1150 RETURN
1200 RESTORE
1215 FORJ=1TO12:READX$:NEXT
1220 READN
1230 FORJ=1TOD
1240 READX$:READY$
1245 X$=X$+Y$:READY$
1250 NEXT J
1260 RETURN
1300 PRINT"[1HOME][16CRSRD][3CRS
RR][1COMM+][1CRSRL]";
1305 A$=""
1307 GETQ$:IFQ$=""THEN 1307
1309 IFASC(Q$)<>20 THEN1314
1310 IFA$=""THEN1307
1311 PRINT"[1CRSRL][1SPC][1CRSRL
]";:A$=LEFT$(A$,LEN(A$)-1):GOTO1
307
1314 IFASC(Q$)=13 THEN 1321
1315 IF Q$<"0" OR (Q$>"9" AND Q$
<"A")OR Q$>"Z" THEN 1307
1316 PRINTQ$;
1317 A$=A$+Q$:GOTO1307
1321 P=LEN(A$):IF P=0THEN1300
1322 IFASC(RIGHT$(A$,1))<>32THEN
1330
1323 A$=MID$(A$,1,P-1)
1326 GOTO1321
1330 IF LEFT$(Y$,1)<>"%"THEN1380

1340 P=2
1350 IFMID$(Y$,P,1)<>"-"THENP=P+
1:GOTO1350
1360 IFVAL(A$)>=VAL(MID$(Y$,2,P-
1))ANDVAL(A$)<=VAL(MID$(Y$,P+1,L
EN(Y$))) THEN1500
1370 R=0:RETURN
1380 P=1
1390 IFMID$(Y$,P,1)="#"THEN1370
1400 Q=P
1410 IFMID$(Y$,Q,1)<>"%"THENQ=Q+
1:GOTO1410
1420 IFMID$(Y$,P,Q-P)=A$THEN1500

1430 P=Q+1:GOTO1390
1500 :R=1:RETURN
1600 RESTORE
1620 S=6*(1-R)+INT(6*RND(1))+1
1630 FORJ=1TOS:READZ$:NEXT
1640 PRINT"[1COMM8][1HOME][17CRS
RD][2CRSRR]";Z$
1650 IFT<=2 THEN GOTO1750
1660 PRINT"[2CRSRR]LA[1SPC]RESPU
ESTA[1SPC]CORRECTA[1SPC]ES":PRIN
T"[2CRSRR]";
1670 IF LEFT$(Y$,1)="%"THEN1710
1680 P=1
1690 IFMID$(Y$,P,1)="%"THENGOTO1
750
1700 PRINTMID$(Y$,P,1);:P=P+1:GO
TO1690
1710 PRINTMID$(Y$,2,LEN(Y$)):GOT
O1750
1750 PRINT"[1HOME][21CRSRD][1CRS
RR][1YEL][3SPC]AHORA[1SPC]PULSE[
1SPC]UNA[1SPC]TECLA":RETURN
2000 X$="ABCD"
2010 INPUTY$
2020 IFX$=Y$THEN2050
2030 PRINT"NO":GOTO2000
2050 PRINT"SI"
2060 STOP
8010 DATASI,FABULOSO!,MARAVILLOS
O!,BIEN,"ASI[1SPC]ES!","CORRECTO
"
8020 DATANO,MAL,PESIMO,FATAL,"NO
[1SPC]ES[1SPC]CORRECTO",INCORREC
TO
9000 DATA11
9010 DATA"[3SPC]£TIGRE£[1SPC]=[1
SPC]£LEON£%%VERDADERO[1SPC]O[1SP
C]FALSO?%%","#","FALSO%F%#"
9020 DATA"[3SPC]5[1SPC]<[1SPC]7%
%VERDADERO[1SPC]O[1SPC]FALSO?%%"
,"#","VERDADERO%V%#"
9030 DATA"[3SPC]7[1SPC]<=[1SPC]3
%%VERDADERO[1SPC]O[1SPC]FALSO?%%
","#","FALSO%F%#"
9040 DATA"[3SPC]£RATON£[1SPC]=[1
SPC]£CABALLO£%%VERDADERO[1SPC]O[
1SPC]FALSO?%%","#","FALSO%F%"
9050 DATA"[3SPC]£UNICORN£[1SPC]<
>[1SPC]7.3%%ESTO[1SPC]ES%%A)[1SP
C]VERDADERO%B)FALSO%C)[1SPC]NO[1
SPC]ES[1SPC]BASIC%%"
9055 DATA"#","C%NO[1SPC]ES[1SPC]
BASIC%#"
9060 DATA"[3SPC]£TRES[1SPC]PUNTO
[1SPC]CINCO£[1SPC]=[1SPC]£3.5£%%
```

```
ESTO[1SPC]ES%A)[1SPC]V%B)[1SPC]F
%C)[1SPC]NO[1SPC]ES"
9065 DATA"[1SPC]BASIC%%#","FALSO
%B%F%#"
9070 DATA"[3SPC]SI[1SPC]Y=37[1SP
C]Y[1SPC]Z=25%[3SPC]Y-3[1SPC]<[1
SPC]Z+5[6SPC]ES%%A)[1SPC]VERDADE
RO%B)[1SPC]FALSO%%","#"
9075 DATAFALSO%B%F%#
9080 DATA"[3SPC]SI[1SPC]X[1SPC]T
IENE[1SPC]EL[1SPC]VALOR[1SPC]73,
%[2SPC]X[1SPC]>=[1SPC]73[1SPC]ES
%%A)[1SPC]V[3SPC]%B)[1SPC]F[4SPC
]%C)[1SPC]ILEGAL"
9085 DATA"%%#",VERDADERO%A%V%#
9090 DATA"[3SPC]£LAMP£=£LAMPOON£
%%[2SPC]VERDADERO[1SPC]O[1SPC]FA
LSO?%",#,FALSO%F%#
9100 DATA"[1SPC]CONSIDERE:%%10[1
SPC]RS=5%20[1SPC]PRINT[1SPC]RS,1
/RS%30[1SPC]RS=RS+2%40[1SPC]IF[1
SPC]RS<17[1SPC]THEN[1SPC]20%"
9105 DATA"50[1SPC]STOP%%[3SPC]CU
ANTOS[1SPC]BUCLES[1SPC]HACE[2SPC
]?%#",6%#
9110 DATA"[2SPC]MIRE:%%10[1SPC]S
J$=£F£%20[1SPC]PRINT[1SPC]SJ$%30
[1SPC]SJ$=£J£+SJ$+£K£%40[1SPC]IF
[1SPC]SJ$[1SPC]<"
9115 DATA"[1SPC]£JJJ£[1SPC]THEN[
1SPC]20%50[1SPC]STOP%CUAL[1SPC]E
S[1SPC]EL[1SPC]ULTIMO[1SPC]VALOR
[1SPC]MOSTRADO?%#",JJFKK%#
```



PROGRAMA: UNI8PROG

```
1 REM COPYRIGHT (C) ANDREW COLIN
 1981
2 REM ESTE PROGRAMA MUESTRA LA T
ABLA DEL SIETE
3 REM PERO CONTIENE VARIOS ERROR
ES DELIBERADOS
10 PRINT "TABLA[1SPC]DEL[1SPC]SI
ETE"
20 PRIMT
30 X=1
40 PRINT"7*";X;"=";7*X50X=X+1
60 IF X>13 THEN40
70 STOP
```

PROGRAMA: UNI9CUEST

```
2 REM UNIT9QUIZ64
3 REM COPYRIGHT (C) ANDREW COLIN
 1982
5 POKE 53281,0
10 PRINT"[1CLR][1CRSRD][1COMM8][
3SPC]PROGRAMA[1SPC]DE[1SPC]PREGU
NTAS[1SPC]Y[1SPC]RESPUESTAS"
20 PRINT:PRINT"[4SPC]CADA[1SPC]V
EZ,PULSE[1SPC]UNA[1SPC]TECLA[1SP
C]PARA[1SPC]"
30 PRINT"[3SPC]PASAR[1SPC]A[1SPC
]LA[1SPC]PROXIMA[1SPC]PREGUNTA"
40 PRINT:PRINT"[6SPC]SIEMPRE[1SP
C]SEGUIDOS[1SPC]"
50 PRINT"[2SPC]LA[1SPC]TECLA[2SP
C][1YEL]RETURN[1COMM8][3SPC]"
60 PRINT:PRINT:PRINT"[3SPC]BUENA
[1SPC]SUERTE"
70 FORJ=1TO12:READY$:NEXT
80 READN
90 DIMZ(N)
100 FORJ=1TON:Z(N)=0:NEXT
110 SC=0
120 FORJ=1TON
130 IFZ(J)>=0THEN 200
140 NEXTJ
150 FORJ=1TO2000:NEXTJ
155 PRINT"[1WHT][1CLR][1CRSRD][7
SPC]SU[1SPC]PUNTUACION";SC
160 PRINT:PRINT"[4SPC]PREGUNTAS[
1SPC]HACERTADAS"
170 PRINT:PRINT"[4SPC]A[1SPC]LA[
1SPC]PRIMERA";N
180 PRINT:PRINT"[4SPC]ESTO[1SPC]
ES";INT(100*SC/N+0.5);"%[1SPC]"
190 END
200 D=1+INT(N*RND(1))
210 IF Z(D)=-1THEN200
220 C=Z(D)+1
230 GOSUB1200
240 FORT=1TO3
250 GETA$:IFA$=""THEN250
260 GOSUB1000
270 GOSUB1300
280 GOSUB1600
285 IFR=1ANDT=1ANDZ(D)=0THEN SC=
SC+1
290 IF R=1 AND T=1 THENZ(D)=-1:G
OTO400
300 IF R=1 THEN 320
310 NEXTT
320 Z(D)=Z(D)+1:GOTO120
400 GOTO120
1000 PRINT"[1CLR]"
1010 PRINT"[1CLR]"
1020 FORJ=0TO39:POKE1024+J,160:P
OKE55296+J,C:NEXT
1030 FORJ=0TO920STEP40:POKE1103+
J,160:POKE55375+J,C:NEXT
1040 FORJ=39TO0STEP-1:POKE1984+J
,160:POKE56256+J,C:NEXT
1050 FORJ=920TO0STEP-40:POKE1064
+J,160:POKE55336+J,C:NEXT
1055 W=2:IFLEN(X$)>110THENW=1
1060 P=1:J=3: K=3
1070 A$=MID$(X$,P,1):P=P+1
1080 IFA$="#"THEN 1150
1090 IFA$="%"THENJ=J+W:K=3:GOTO1
070
1095 IFA$="£"THEN A$=CHR$(34)
1100 POKE1024+40*J+K,ASC(A$)AND
63:POKE55296+40*J+K,15:K=K+1
1110 GOTO1070
1150 RETURN
1200 RESTORE
1215 FORJ=1TO12:READX$:NEXT
1220 READN
1230 FORJ=1TOD
1240 READX$:READY$
1245 X$=X$+Y$:READY$
1250 NEXT J
1260 RETURN
1300 PRINT"[1HOME][16CRSRD][3CRS
RR][1COMM+][1CRSRL]";
1305 A$=""
1307 GETQ$:IFQ$=""THEN 1307
1309 IFASC(Q$)<>20 THEN1314
1310 IFA$=""THEN1307
1311 PRINT"[1CRSRL][1SPC][1CRSRL
]";:A$=LEFT$(A$,LEN(A$)-1):GOTO1
307
1314 IFASC(Q$)=13 THEN 1321
1315 IF Q$<"0" OR (Q$>"9" AND Q$
<"A")OR Q$>"Z" THEN 1307
1316 PRINTQ$;
1317 A$=A$+Q$:GOTO1307
1321 P=LEN(A$):IF P=0THEN1300
1322 IFASC(RIGHT$(A$,1))<>32THEN
1330
1323 A$=MID$(A$,1,P-1)
1326 GOTO1321
1330 IF LEFT$(Y$,1)<>"%"THEN1380

1340 P=2
1350 IFMID$(Y$,P,1)<>"-"THENP=P+
1:GOTO1350
1360 IFVAL(A$)>=VAL(MID$(Y$,2,P-
1))ANDVAL(A$)<=VAL(MID$(Y$,P+1,L
EN(Y$))) THEN1500
1370 R=0:RETURN
1380 P=1
1390 IFMID$(Y$,P,1)="#"THEN1370
1400 Q=P
1410 IFMID$(Y$,Q,1)<>"%"THENQ=Q+
1:GOTO1410
```

```
1420 IFMID$(Y$,P,Q-P)=A$THEN1500

1430 P=Q+1:GOTO1390
1500 :R=1:RETURN
1600 RESTORE
1620 S=6*(1-R)+INT(6*RND(1))+1
1630 FORJ=1TOS:READZ$:NEXT
1640 PRINT"[1COMM8][1HOME][17CRS
RD][2CRSRR]";Z$
1650 IFT<=2 THEN GOTO1750
1660 PRINT"[2CRSRR]LA[1SPC]RESPU
ESTA[1SPC]CORRECTA[1SPC]ES":PRIN
T"[2CRSRR]";
1670 IF LEFT$(Y$,1)="%"THEN1710
1680 P=1
1690 IFMID$(Y$,P,1)="%"THENGOTO1
750
1700 PRINTMID$(Y$,P,1);:P=P+1:GO
TO1690
1710 PRINTMID$(Y$,2,LEN(Y$)):GOT
O1750
1750 PRINT"[1HOME][21CRSRD][1CRS
RR][1YEL][3SPC]PULSE[1SPC]UNA[1S
PC]TECLA[1SPC]":RETURN
2000 X$="ABCD"
2010 INPUTY$
2020 IFX$=Y$THEN2050
2030 PRINT"NO":GOTO2000
2050 PRINT"SI[1SPC]"
2060 STOP
8010 DATASI,CORRECTO,BIEN ,VALE,
"ES[1SPC]ASI",OK!!!!!!!!
8020 DATANO,MAL,"NO[1SPC]ES[1SPC
]ASI",FATAL,"SI[1SPC]SIGUE[1SPC]
ASI[1SPC]VA[1SPC]MUY[1SPC]MAL[1S
PC]VALE",INCORRECTISIMO
9000 DATA7
9010 DATA"[1SPC]SI[1SPC]SU[1SPC]
ORDENADOR[1SPC]ENTRA[1SPC]EN[1SP
C]UN%BUCLE[1SPC]INFINITO%QUE[1SP
C]HARA?%%
9015 DATA"A)[1SPC]PULSAR[1SPC]RU
N/STOP%B)[1SPC]PULSAR[1SPC]RESTO
RE%C)[1SPC]APAGAR[1SPC]LA[1SPC]M
AQUINA%#",A%#
9020 DATA"[2SPC]LA[1SPC]TECLA[1S
PC]RESTORE[1SPC]PARA[1SPC]EL%PRO
GRAMA?%%",#,NO%N%#
9030 DATA"[1SPC]DE[1SPC]QUE[1SPC
]FUNCION[1SPC]ES[1SPC]EL[1SPC]CO
DIGO%E[1SPC]EN[1SPC]REVERSE?%%A)
[1SPC]CTRL[1SPC]Y[1SPC]WHT%B)[1S
PC]CTRL[1SPC]"
9035 DATA"Y[1SPC]GRN%C)[1SPC]SHI
FT[1SPC]Y[1SPC]X%D)[1SPC]SHIFT[1
SPC]Y[1SPC]E%%#","CTRL[1SPC]Y[1S
PC]WHT%A%#"
9040 DATA"[3SPC]ES[1SPC]LEGAL[1S
PC]INCLUIR[1SPC]UN%COMANDO[1SPC]
COMO%%150[1SPC]TI$[1SPC]=[1SPC]£
090000£%%EN[1SPC]UN[1SPC]PRO"
9045 DATA"GRAMA?%#",SI%S%#
9050 DATA"[3SPC]CUAL[1SPC]DE[1SP
C]ESTAS[1SPC]OPERACIONES%APAGA[1
SPC]EL[1SPC]RELOJ[1SPC]?%%A)LA[1
SPC]TECLA[1SPC]RUN/STOP"
9055 DATA"[1SPC]Y[1SPC]RESTORE%C
)CARGAR[1SPC]UN[1SPC]NUEVO[1SPC]
PROGRAMA%D)DESCONECTAR%E)NINGUNA
%#"
9057 DATA"DESCONECTAR%D%#
9060 DATA"[1SPC]CUANDO[1SPC]PULS
E[1SPC]RUN/STOP[1SPC]Y%RESTORE[1
SPC]LA[1SPC]PANTALLA[1SPC]SE%LIM
PIA.QUE[1SPC]PASA[1SPC]CON[1SPC]
SU
9065 DATA"PROGRAMA?%%A)LO[1SPC]L
ISTA%B)LO[1SPC]REESCRIBE%C)ALGUN
A[1SPC]OTRA[1SPC]COSA%D)NO[1SPC]
PUEDE#
9067 DATA"LISTARLO%A%#
9070 DATA "QUE[1SPC]ARA[1SPC]EL[
1SPC]RELOJ[1SPC]A[1SPC]%MEDIA[1S
PC]NOCHE?%%A)[1SPC]CUANDO[1SPC]M
ARCA[1SPC]LAS[1SPC]£235959£%B)"
9075 DATA"[1SPC]PASA[1SPC]A£2400
00£%C)[1SPC]PASA[1SPC]A[1SPC]£00
0000£%#",C%#
9080 DATA"[3SPC]SI[1SPC]X[1SPC]T
IENE[1SPC]EL[1SPC]VALOR[1SPC]73,
%[2SPC]X[1SPC]>=[1SPC]73[1SPC]IS
%%A)[1SPC]V%B)[1SPC]F%C)[1SPC]IL
EGAL"
9085 DATA"%%#",V%A%V%#
9090 DATA"[3SPC]£LAMP£=£LAMPOON£
%%[2SPC]V[1SPC]O[1SPC]F?%",#,F%F
%#
9100 DATA"[1SPC]CONSIDERE:%%10[1
SPC]RS=5%20[1SPC]PRINT[1SPC]RS.1
/RS%30[1SPC]RS=RS+2%40[1SPC]IF[1
SPC]RS<17[1SPC]THEN[1SPC]20%"
9105 DATA"50[1SPC]STOP%%[2SPC]CU
ANTOS[1SPC]BUCLES[1SPC]HACE?%#",
6%#
9110 DATA"[2SPC]MIRE!!!:%%10[1SP
C]SJ$=£F£%20[1SPC]PRINT[1SPC]SJ$
%30[1SPC]SJ$=£J£+SJ$+£K£%40[1SPC
]IF[1SPC]SJ$[1SPC]<"
9115 DATA"[1SPC]£JJJ£[1SPC]THEN[
1SPC]20%50[1SPC]STOP%CUAL[1SPC]E
S[1SPC]EL[1SPC]ULTIMO[1SPC]VALOR
[1SPC]MOSTRADO?%#",JJFKK%#
```

PROGRAMA: UNI10CUEST

```
5 POKE 53281,0
6 POKE 53280,0
10 PRINT"[1CLR][1CRSRD][1WHT][3S
PC]PROGRAMA[1SPC]DE[1SPC]PREGUNT
AS[1SPC]Y[1SPC]RESPUESTAS"
20 PRINT:PRINT"[3SPC]CADA[1SPC]V
EZ[1SPC]PULSE[1SPC]UNA[1SPC]TECL
A[2SPC]"
30 PRINT"[2SPC]PARA[1SPC]PASAR[1
SPC]A[1SPC]LA[1SPC]SIGUIENTE[1SP
C]PREGUNTA"
40 PRINT:PRINT"[5SPC]TERMINE[1SP
C]SIEMPRE[1SPC]SUS[1SPC]RESPUEST
AS"
50 PRINT"[2SPC]CON[1SPC]LA[1SPC]
TECLA[1SPC][1YEL]RETURN[1COMM8]"

60 PRINT:PRINT:PRINT"[2SPC]!BUEN
A[1SPC]SUERTE!"
70 FORJ=1TO12:READY$:NEXT
80 READN
90 DIMZ(N)
100 FORJ=1TON:Z(N)=0:NEXT
110 SC=0
120 FORJ=1TON
130 IFZ(J)>=0THEN 200
140 NEXTJ
150 FORJ=1TO2000:NEXTJ
155 PRINT"[1WHT][1CLR][1CRSRD][6
SPC]SU[1SPC]PUNTUACION[1SPC]ES";
SC
160 PRINT:PRINT"[4SPC]RESPUESTAS
[1SPC]CORRECTAS"
170 PRINT:PRINT"[4SPC]A[1SPC]LA[
1SPC]PRIMERA[1SPC]DE[1SPC]UN[1SP
C]TOTAL[1SPC]DE";N
180 PRINT:PRINT"[4SPC]ESTO[1SPC]
ES";INT(100*SC/N+0.5);"%"
190 END
200 D=1+INT(N*RND(1))
210 IF Z(D)=-1THEN200
220 C=Z(D)+1
230 GOSUB1200
240 FORT=1TO3
250 GETA$:IFA$=""THEN250
260 GOSUB1000
270 GOSUB1300
280 GOSUB1600
285 IFR=1ANDT=1ANDZ(D)=0THEN SC=
SC+1
290 IF R=1 AND T=1 THENZ(D)=-1:G
OTO400
300 IF R=1 THEN 320
310 NEXTT
320 Z(D)=Z(D)+1:GOTO120
400 GOTO120
1000 PRINT"[1CLR]"
1010 PRINT"[1CLR]"
1020 FORJ=0TO39:POKE1024+J,160:P
OKE55296+J,C:NEXT
1030 FORJ=0TO920STEP40:POKE1103+
J,160:POKE55375+J,C:NEXT
1040 FORJ=39TO0STEP-1:POKE1984+J
,160:POKE56256+J,C:NEXT
1050 FORJ=920TO0STEP-40:POKE1064
+J,160:POKE55336+J,C:NEXT
1055 W=2:IFLEN(X$)>110THENW=1
1060 P=1:J=3: K=3
1070 A$=MID$(X$,P,1):P=P+1
1080 IFA$="#"THEN 1150
1090 IFA$="%"THENJ=J+W:K=3:GOTO1
070
1095 IFA$="£"THEN A$=CHR$(34)
1100 POKE1024+40*J+K,ASC(A$)AND
63:POKE55296+40*J+K,15:K=K+1
1110 GOTO1070
1150 RETURN
1200 RESTORE
1215 FORJ=1TO12:READX$:NEXT
1220 READN
1230 FORJ=1TOD
1240 READX$:READY$
1245 X$=X$+Y$:READY$
1250 NEXT J
1260 RETURN
1300 PRINT"[1HOME][16CRSRD][3CRS
RR][1COMM+][1CRSRL]";
1305 A$=""
1307 GETQ$:IFQ$=""THEN 1307
1309 IFASC(Q$)<>20 THEN1314
1310 IFA$=""THEN1307
1311 PRINT"[1CRSRL][1SPC][1CRSRL
]";:A$=LEFT$(A$,LEN(A$)-1):GOTO1
307
1314 IFASC(Q$)=13 THEN 1321
1315 IF Q$<"0" OR (Q$>"9" AND Q$
<"A")OR Q$>"Z" THEN 1307
1316 PRINTQ$;
1317 A$=A$+Q$:GOTO1307
1321 P=LEN(A$):IF P=0THEN1300
1322 IFASC(RIGHT$(A$,1))<>32THEN
1330
1323 A$=MID$(A$,1,P-1)
1326 GOTO1321
1330 IF LEFT$(Y$,1)<>"%"THEN1380

1340 P=2
1350 IFMID$(Y$,P,1)<>"-"THENP=P+
1:GOTO1350
1360 IFVAL(A$)>=VAL(MID$(Y$,2,P-
1))ANDVAL(A$)<=VAL(MID$(Y$,P+1,L
EN(Y$))) THEN1500
1370 R=0:RETURN
1380 P=1
1390 IFMID$(Y$,P,1)="#"THEN1370
```

```
1400 Q=P
1410 IFMID$(Y$,Q,1)<>"%"THENQ=Q+
1:GOTO1410
1420 IFMID$(Y$,P,Q-P)=A$THEN1500

1430 P=Q+1:GOTO1390
1500 :R=1:RETURN
1600 RESTORE
1620 S=6*(1-R)+INT(6*RND(1))+1
1630 FORJ=1TOS:READZ$:NEXT
1640 PRINT"[1COMM8][1HOME][17CRS
RD][2CRSRR]";Z$
1650 IFT<=2 THEN GOTO1750
1660 PRINT"[2CRSRR]LA[1SPC]RESPU
ESTA[1SPC]CORRECTA[1SPC]ES":PRIN
T"[2CRSRR]";
1670 IF LEFT$(Y$,1)="%"THEN1710
1680 P=1
1690 IFMID$(Y$,P,1)="%"THENGOTO1
750
1700 PRINTMID$(Y$,P,1);:P=P+1:GO
TO1690
1710 PRINTMID$(Y$,2,LEN(Y$)):GOT
O1750
1750 PRINT"[1HOME][21CRSRD][1CRS
RR][1YEL][3SPC]AHORA[1SPC]PULSE[
1SPC]UNA[1SPC]TECLA":RETURN
2000 X$="ABCD"
2010 INPUTY$
2020 IFX$=Y$THEN2050
2030 PRINT"NO":GOTO2000
2050 PRINT"SI"
2060 STOP
8010 DATASI,CORRECTO,BIEN,FABULO
SO,"!MUY[1SPC]BIEN!",OK
8020 DATANO,MAL,"MUY[1SPC]MAL",N
O ES ASI,"FATAL","CASI[1SPC]PERO
[1SPC]NO"
9000 DATA9
9010 DATA"EL[1SPC]COMANDO[1SPC]I
NPUT[1SPC]PUEDE[1SPC]SER[1SPC]US
ADO%A)PARA[1SPC]NUMEROS%B)PARA[1
SPC]STRINGS%C)[1SPC]PARA[1SPC]NU

9015 DATA"MEROS[1SPC]Y[1SPC]STRI
NGS[1SPC]JUNTOS%D)[1SPC]PARA[1SP
C]NADA%%#",C%#
9020 DATA"INPUT£XXX£;J$%QUE[1SPC
]LONGITUD[1SPC](EN[1SPC]CARACTER
ES)[1SPC]PUEDE%TENER[1SPC]EL[1SP
C]STRING[1SPC]J$?%"
9025 DATA#,"38%#"
9030 DATA"[2SPC]EL[1SPC]COMANDO%
[2SPC]INPUT£RQ$£;RS%ENTRA%A)[1SP
C]RQ$[1SPC]SOLO%B)[1SPC]RS[1SPC]
SOLO%C)[1SPC]JUNTOS
9035 DATA"[1SPC]RQ$[1SPC]Y[1SPC]
RS%D)[1SPC]NINGUNO%#","RS%B%#"
9040 DATA"[3SPC]INPUT[1SPC]X,Y%E
SPERA%%A)[1SPC]1[1SPC]VALOR%B)[1
SPC]2[1SPC]VALORES%C)[1SPC]MAS[1
SPC]DE[1SPC]2[1SPC]VALORES%%"
9045 DATA#,DOS%2%B%#
9050 DATA"'INPUT£EDAD£;D'[1SPC]S
I[1SPC]PULSA[1SPC]%£PEPE£.QUE[1S
PC]PASA?%%A)EL[1SPC]VAL
9055 DATAOR SE ACEPTA%B)EL 64 DI
CE%'REDO FROM START'%C)EL PROGRA
MA FALLA%#,B%#
9060 DATA"INPUT£NOMBRE£;DD$%SI[1
SPC]ESCRIBIMOS%£JUAN£%QUE[1SPC]P
ASA?%%A)EL[1SPC]VAL
9065 DATAOR SE ACEPTA%B)EL 64 DI
CE%'REDO FROM START'%C)EL PROGRA
MA FALLA%#,A%#
9070 DATA"[3SPC]10[1SPC]INPUT[1S
PC]777%%ES[1SPC]%A)[1SPC]UN[1SPC
]COMANDO[1SPC]ILEGAL%B)[1SPC]UN[
1SPC]COMANDO[1SPC]VALIDO%C)[1SPC
]UN
9075 DATA"[1SPC]COMANDO[1SPC]EN[
1SPC]DESUSO%%#",A%#
9080 DATA"10[1SPC]INPUT[1SPC]T%2
0[1SPC]T=93%%ES[1SPC]ESTA[1SPC]S
ECUENCIA[1SPC]RAZONABLE?%%",#,"N
O%N%#
9090 DATA"SI[1SPC]SE[1SPC]PIERDE
[1SPC]EL[1SPC]CONTROL%[1SPC]COMO
[1SPC]SE[1SPC]RECUPERA?%%A)C
9095 DATAON LA TECLA RESTORE%B)
RUN/STOP Y RESTORE%C) CRTL Y RES
TORE%#,B%#
```

PROGRAMA: UNI11PROG

```
0 REM UNI11PROG 64
1 PRINT"[1CLR][1WHT]"
10 POKE53280,0:POKE53281,0
20 PRINT"CUAL[1SPC]ES[1SPC]SU[1S
PC]"
30 INPUT"[1SPC]NOMBRE";X$
40 PRINT
50 PRINT"HOLA[2SPC]";X$
60 PRINT
70 PRINT"POR[1SPC]FAVOR[1SPC]RES
PONDA[1SPC]ESTAS"
80 PRINT"SIMPLES[1SPC]PREGUNTAS"

90 SC=0
100 PRINT
110 PRINT"CUAL[1SPC]ES[1SPC]LA[1
SPC]CAPITAL"
120 INPUT"DE[1SPC]FRANCIA";A$
130 IF A$="PARIS" THEN160
140 PRINT"MAL.[1SPC]ES[1SPC]PARI
S"
150 GOTO190
160 PRINT"!BIEN!"
170 PRINT "OTRA[1SPC]VEZ[1SPC]";
X$
180 SC=SC+1
190 PRINT
200 PRINT"QUIEN[1SPC]DESCUBRIO"
210 INPUT"AMERICA";A$
220 IFA$="COLON"THEN270
230 PRINT"PERDON[1SPC]";X$
240 PRINT"HA[1SPC]HABIDO[1SPC]UN
[1SPC]FALLO[1SPC]ERA"
250 PRINT"COLON[3SPC]"
260 GOTO300
270 PRINT"CORRECTO.[1SPC]";X$;"[
1SPC]ES"
280 PRINT"UN[1SPC]EXPERTO[1SPC]H
ISTORIADOR"
290 SC=SC+1
300 PRINT
305 IF SC=2THEN390
310 IF SC=1 THEN350
320 PRINT"NO[1SPC]HA[1SPC]HACERT
ADO[1SPC]NINGUNA"
330 PRINT"PREGUNTA[1SPC]A[1SPC]L
A[1SPC]PRIMERA"
340 GOTO370
350 PRINT"SOLO[1SPC]HA[1SPC]ACER
TADO[1SPC]UNA"
360 PRINT"PREGUNTA[1SPC]A[1SPC]L
A[1SPC]PRIMERA."
370 PRINT"PRUEBE[1SPC]OTRA[1SPC]
VEZ."
380 GOTO90
390 PRINT"HA[1SPC]SIDO[1SPC]EXCE
LENTE,"
400 PRINTX$;".[1SPC]TODAS[1SPC]L
AS"
410 PRINT"RESPUESTAS[1SPC]CORREC
TAS"
420 STOP
```

PROGRAMA: UNI12CUEST

```
0 REM UNI12CUEST 64
1 RESTORE
5 POKE 53281,0
6 POKE 53280,0
10 PRINT"[1CLR][1WHT][2SPC]PROGR
AMA[1SPC]DE[1SPC]PREGUNTAS[1SPC]
Y[1SPC]RESPUESTAS"
20 PRINT:PRINT"[4SPC]PULSE[1SPC]
UNA[1SPC]TECLA[1SPC]PARA"
30 PRINT"[1SPC]PASAR[1SPC]A[1SPC
]LA[1SPC]SIGUIENTE[1SPC]PREGUNTA
"
40 PRINT:PRINT"[1SPC]SIEMPRE[1SP
C]SEGUIDAS[1SPC]SUS[1SPC]RESPUES
TAS"
50 PRINT"[1SPC]CON[1SPC]LA[1SPC]
TECLA[1SPC][1YEL]RETURN[1COMM8]"

60 PRINT:PRINT:PRINT"[2SPC]!BUEN
A[1SPC]SUERTE!"
70 FORJ=1TO13:READY$:NEXT
80 N=9
90 DIMZ(N)
100 FORJ=1TON:Z(N)=0:NEXT
110 SC=0
120 FORJ=1TON
130 IFZ(J)>=0THEN 200
140 NEXTJ
150 FORJ=1TO2000:NEXTJ
155 PRINT"[1WHT][1CLR][1CRSRD][5
SPC]SU[1SPC]PUNTUACION[1SPC]ES[1
SPC]";SC
160 PRINT:PRINT"[4SPC]PREGUNTAS[
1SPC]ACERTADAS[1SPC]A"
170 PRINT:PRINT"[4SPC]LA[1SPC]PR
IMERA";N
180 PRINT:PRINT"[4SPC]ESTO[1SPC]
ES";INT(100*SC/N+0.5);"%[5SPC]"
190 END
200 D=1+INT(N*RND(1))
210 IF Z(D)=-1THEN200
220 C=Z(D)+1
230 GOSUB1200
240 FORT=1TO3
250 GETA$:IFA$=""THEN250
260 GOSUB1000
270 GOSUB1300
280 GOSUB1600
285 IFR=1ANDT=1ANDZ(D)=0THEN SC=
SC+1
290 IF R=1 AND T=1 THENZ(D)=-1:G
OTO400
300 IF R=1 THEN 320
310 NEXTT
320 Z(D)=Z(D)+1:GOTO120
400 GOTO120
1000 PRINT"[1CLR]"
1010 PRINT"[1CLR]"
1020 FORJ=0TO39:POKE1024+J,160:P
OKE55296+J,C:NEXT
1030 FORJ=0TO920STEP40:POKE1103+
J,160:POKE55375+J,C:NEXT
1040 FORJ=39TO0STEP-1:POKE1984+J
,160:POKE56256+J;C:NEXT
1050 FORJ=920TO0STEP-40:POKE1064
+J,160:POKE55336+J,C:NEXT
1055 W=2:IFLEN(X$)>110THENW=1
1060 P=1:J=3: K=3
1070 A$=MID$(X$,P,1):P=P+1
1080 IFA$="#"THEN 1150
1090 IFA$="%"THENJ=J+W:K=3:GOTO1
070
1095 IFA$="£"THEN A$=CHR$(34)
1100 POKE1024+40*J+K,ASC(A$)AND
63:POKE55296+40*J+K,15:K=K+1
1110 GOTO1070
1150 RETURN
1200 RESTORE
1215 FORJ=1TO13:READX$:NEXT
1230 FORJ=1TOD
1240 READX$:READY$
1245 X$=X$+Y$:READY$
1250 NEXT J
1260 RETURN
1300 PRINT"[1HOME][16CRSRD][3CRS
RR][1COMM+][1CRSRL]";
1305 A$=""
1307 GETQ$:IFQ$=""THEN 1307
1309 IFASC(Q$)<>20 THEN1314
1310 IFA$=""THEN1307
1311 PRINT"[1CRSRL][1SPC][1CRSRL
]";:A$=LEFT$(A$,LEN(A$)-1):GOTO1
307
1314 IFASC(Q$)=13 THEN 1321
1315 IF Q$<"0" OR (Q$>"9" AND Q$
<"A")OR Q$>"Z" THEN 1307
1316 PRINTQ$;
1317 A$=A$+Q$:GOTO1307
1321 P=LEN(A$):IF P=0THEN1300
1322 IFASC(RIGHT$(A$,1))<>32THEN
1330
1323 A$=MID$(A$,1,P-1)
1326 GOTO1321
1330 IF LEFT$(Y$,1)<>"%"THEN1380

1340 P=2
1350 IFMID$(Y$,P,1)<>"-"THENP=P+
1:GOTO1350
1360 IFVAL(A$)>=VAL(MID$(Y$,2,P-
1))ANDVAL(A$)<=VAL(MID$(Y$,P+1,L
EN(Y$))) THEN1500
1370 R=0:RETURN
1380 P=1
1390 IFMID$(Y$,P,1)="#"THEN1370
1400 Q=P
```

```
1410 IFMID$(Y$,Q,1)<>"%"THENQ=Q+
1:GOTO1410
1420 IFMID$(Y$,P,Q-P)=A$THEN1500

1430 P=Q+1:GOTO1390
1500 :R=1:RETURN
1600 RESTORE
1620 S=6*(1-R)+INT(6*RND(1))+1
1630 FORJ=1TOS:READZ$:NEXT
1640 PRINT"[1COMM8][1HOME][17CRS
RD][2CRSRR]";Z$
1650 IFT<=2 THEN GOTO1750
1660 PRINT"[2CRSRR]LA[1SPC]RESPU
ESTA[1SPC]CORRECTA[1SPC]ES[1SPC]
":PRINT"[2CRSRR]";
1670 IF LEFT$(Y$,1)="%"THEN1710
1680 P=1
1690 IFMID$(Y$,P,1)="%"THENGOTO1
750
1700 PRINTMID$(Y$,P,1);:P=P+1:GO
TO1690
1710 PRINTMID$(Y$,2,LEN(Y$)):GOT
O1750
1750 PRINT"[1HOME][21CRSRD][1CRS
RR][1YEL][3SPC]AHORA[1SPC]PULSE[
1SPC]UNA[1SPC]TECLA":RETURN
2000 X$="ABCD"
2010 INPUTY$
2020 IFX$=Y$THEN2050
2030 PRINT"NO":GOTO2000
2050 PRINT"SI"
2060 STOP
8010 DATASI,CORRECTO,BIEN,FABULO
SO,"ES[1SPC]ASI",BRAVO,OK
8021 DATA NO
8022 DATA MAL
8023 DATA INCORRECTO
8024 DATA FATAL
8025 DATA NO ES ASI
8026 DATA MUY MAL
9010 DATA"[3SPC]PUEDE[1SPC]UTILI
ZARSE[1SPC]A$[1SPC]PARA[1SPC]EL%
CONTROL[1SPC]EN[1SPC]UN[1SPC]FOR
-NEXT?%",#
9015 DATANO%N%#
9020 DATA"[1SPC]PUEDE[1SPC]HACER
SE[1SPC]%%10[1SPC]FOR[1SPC]I=T[1
SPC]TO[1SPC]Q[1SPC]%20[1SPC]NEXT
[1SPC]I
9025 DATA"%30[1SPC]STOP%%[1SPC]E
S[1SPC]CORRECTO?[1SPC]%%#",SI%S%
#
9030 DATA"[1SPC]PUEDE[1SPC]EVITA
RSE[1SPC]EL[1SPC]£STEP£[1SPC]EN[
1SPC]UN%%COMANDO[1SPC]FOR-NEXT?%
%A)[1SPC]NUNCA
9035 DATA"%B)[1SPC]SI[1SPC]EL[1S
PC]PASO[1SPC]ES[1SPC]1%C)[1SPC]C
UANDO[1SPC]EL[1SPC]PASO[1SPC]ES[
1SPC]%NEGATIVO%%#",B%#
9040 DATA"VEA%10[1SPC]FOR[1SPC]Q
=20[1SPC]TO[1SPC]37%20[1SPC]PRIN
T[1SPC]Q*Q/5%30[1SPC]NEXT[1SPC]W
%[3SPC]ES[1SPC]ESTO%A)[1SPC]CO
9045 DATA"RRECTO[1SPC]%B)[1SPC]I
NCORRECTO[1SPC]%C)[1SPC]CODIGO[1
SPC]AMBIGUO%%#",INCORRECTO%B%#
9050 DATA"[3SPC]CUANTOS[1SPC]COM
ANDOS[1SPC]PUEDE[1SPC]PONER%%EN[
1SPC]EL[1SPC]CUERPO[1SPC]DE[1SPC
]UN[1SPC]BUCLE[1SPC]£FOR£?%%A)
9055 DATA"[1SPC]0[1SPC]O[1SPC]MA
S%B)[1SPC]1[1SPC]O[1SPC]MAS%C)[1
SPC]1[1SPC]%%#",A%#
9060 DATA"[2SPC]PUEDE[1SPC]EL[1S
PC]VALOR[1SPC]INICIAL[1SPC]DE[1S
PC]UN%%BUCLE[1SPC]£FOR£%%SER[1SP
C]UNA
9065 DATA"[1SPC]VARIABLE[1SPC]SO
LA?%%(COMO[3SPC]£FOR[1SPC]R[1SPC
]=[1SPC]Q[1SPC]TO[1SPC]100[1SPC]
STEP[1SPC]4)%%#",SI%S%#
9070 DATA"[1SPC]PUEDE[1SPC]SER[1
SPC]EL[1SPC]PASO[1SPC]UNA[1SPC]V
ARIABLE%[2SPC]EN[1SPC]UN[1SPC]BU
CLE[1SPC]£FOR£[1SPC]-"
9075 DATA"[1SPC]£NEXT£[1SPC]?%#"
,"SI%S%#"
9080 DATA"FIJESE[2SPC]%10[1SPC]F
OR[1SPC]Q=3[1SPC]TO[1SPC]15[1SPC
]STEP[1SPC]4%20[1SPC]PRINT[1SPC]
Q%30[1SPC]NEXT[1SPC]Q%CUANTOS[1S
PC]BUCLE"
9085 DATA"S[1SPC]HACE[1SPC]EL%CU
ERPO[1SPC]DEL[1SPC]FOR-NEXT?%%#"
,"CUATRO%4%#"
9090 DATA"MIRE[4SPC]%%10[1SPC]FO
R[1SPC]R=7[1SPC]TO[1SPC]3%20[1SP
C]PRINT[1SPC]R%30[1SPC]NEXT[1SPC
]R%%CUANTOS[1SPC]BUCLES[1SPC]EFE
CTUA
9095 DATA"[1SPC]EL[1SPC]CUERPO%D
EL[1SPC]FOR-NEXT?%%A)[1SPC]UNO%B
)[1SPC]CINCO%C)[1SPC]NINGUNO[1SP
C]%%#",UNO%1%A%#
```

PROGRAMA: GENSON

```
0 REM GENERADOR EXPERIMENTAL DE
SONIDO (UNIDAD 13) 64
1 POKE53280,0:POKE53281,0
10 REM COPYRIGHT (C) ANDREW COLI
N 1982
15 V=54272:FORI=0TO28:POKEV+I,0:
NEXT
20 PRINT"[1CLR][1WHT]";
30 POKE54295,0:POKE54296,15
40 DIM T$(4)
50 FORJ=1TO4:READ T$(J):NEXT J
60 DATA TRIANGULAR,D. DE SIERRA,
PULSO,RUIDO
70 F=512:P=200:W=1:A=0:D=9:S=0:T
=250
80 PRINT"[1CLR][2CRSRD][1WHT][3C
RSRR]GENERADOR[1SPC]EXPERIMENTAL
[1SPC]DE[1SPC]SONIDO"
90 PRINT"[3CRSRR][32SHIFE]"
100 PRINT:PRINT"[1CRSRR]PULSE[1S
PC]ESPACIO[1SPC]PARA[1SPC]GENERA
R[1SPC]SONIDO":PRINT:
110 PRINT"[1CRSRR][3SPC]O[1SPC]P
ULSE[1SPC]LA[1SPC]LETRA[1SPC]PAR
A[1SPC]CAMBIO":PRINT
120 PRINT"[3CRSRR]PARAMETRO[7SPC
]LETRA[3SPC]VALOR"
125 PRINT"[2CRSRR][25SPC]ACTUAL"
:PRINT
130 PRINT"[3CRSRR]FRECUENCIA[7SP
C](F)[1SPC]=[1SPC]";F
140 PRINT"[3CRSRR]DURACION[1SPC]
TIEMPO[2SPC](T)[1SPC]=[1SPC]";T
150 PRINT"[3CRSRR]SOSTENIMIENTO[
4SPC](S)[1SPC]=[1SPC]";S
160 PRINT"[3CRSRR]ATAQUE[11SPC](
A)[1SPC]=[1SPC]";A
170 PRINT"[3CRSRR]DECAIMIENTO[6S
PC](D)[1SPC]=[1SPC]";D
180 PRINT"[3CRSRR]FORMA[1SPC]DE[
1SPC]ONDA[4SPC](W)[1SPC]ES[1SPC]
";T$(W)
190 IF W=3 THEN  PRINT"[3CRSRR]A
NCHO[1SPC]DE[1SPC]PULSO[3SPC](P)
[1SPC]=[1SPC]";P
200 GET A$:IFA$=""THEN 200
205 IFA$="[1SPC]"THEN Z=0:GOTO61
0
207 Z=1
210 IF A$ <> "F" THEN 250
220 INPUT"[3CRSRR]NUEVA[1SPC]FRE
CUENCIA[1SPC](20-4000)[1SPC]";F
230 IF F<20 OR F>4095  THEN GOSU
B1000:GOTO200
240 GOTO600
250 IF A$<>"T"THEN 290
260 INPUT "[3CRSRR]NUEVA[1SPC]DU
RACION[1SPC](1[1SPC]-[1SPC]5000)
";T
270 IF T<1 OR T>5000 THEN GOSUB1
000:GOTO260
280 GOTO610
290 IFA$<>"S"THEN 330
300 INPUT"[3CRSRR]NUEVO[1SPC]SOS
TENIMIENTO[1SPC](0[1SPC]A[1SPC]1
5)";S
310 IF S<0 OR S>15 THEN GOSUB100
0:GOTO300
320 GOTO610
330 IFA$<>"A"THEN 370
340 INPUT"[3CRSRR]NUEVO[1SPC]ATA
QUE[5SPC](0[1SPC]A[1SPC]15)[1SPC
]";A
350 IF A<0 OR A>15 THEN GOSUB100
0:GOTO340
360 GOTO610
370 IFA$<>"D"THEN 410
380 INPUT"[3CRSRR]NUEVO[1SPC]DEC
AIMIENTO[1SPC](0[1SPC]A[1SPC]15)
[1SPC]";D
390 IF D<0 OR D>15 THEN GOSUB100
0:GOTO380
400 GOTO610
410 IF A$<>"W"THEN 480
420 PRINT"[3CRSRR]NUEVA[1SPC]FOR
MA[2SPC](T,S,P[1SPC]O[1SPC]N)";
425 GETA$:IFA$=""THEN 425
430 IFA$="T"THEN W=1:GOTO610
440 IFA$="S"THEN W=2:GOTO610
450 IFA$="P"THEN W=3:GOTO610
460 IFA$="N"THEN W=4:GOTO610
470 GOSUB1000:PRINT:GOTO420
480 IFA$<>"P"OR W <> 3 THEN 520
490 INPUT"[3CRSRR]NUEVO[1SPC]ANC
HO[1SPC]PUL.[1SPC](8[1SPC]A[1SPC
]999)[1SPC]";P
500 IF P<0 OR P>999THEN GOSUB100
0:GOTO490
510 GOTO610
520 IF A$<>"[1SPC]"THEN GOSUB100
0:GOTO80
600 REM TOCAR NOTA
610 Q=16*F:POKE54273,Q/256:POKE5
4272,(Q-32768)AND255
620 POKE 54275,P/8:POKE54277,16*
A+D:POKE54278,16*S
630 POKE54276,1+2↑(W+3)
640 FORN=1TOT:NEXTN
650 POKE54276,0
660 IF Z=1THEN 80
670 GOTO200
1000 REM ERROR PELIGRO
1010 POKE 54278,0:POKE 54277,8
1020 POKE 54276,0:POKE54276,17
1030 FORNN=100TO20STEP-1.5
```

```
1040 POKE54273,NN
1050 NEXTNN
1060 POKE 54276,0
1070 RETURN
```

PROGRAMA: CARA/CRUZ

```
0 REM CARA/CRUZ 64
1 POKE53280,0:POKE53281,0:PRINT"
[1CLR][1WHT]"
10 H=0
20 T=0
30 PRINT"PULSAR[1SPC]-H-[1SPC]PA
RA[1SPC]CARA"
40 PRINT"[7SPC]-T-[1SPC]PARA[1SP
C]CRUZ"
50 PRINT"[7SPC]-E-[1SPC]PARA[1SP
C]FIN"
60 INPUT"EL[1SPC]PROXIMO[1SPC]ES
";I$
70 IF I$="E"THEN 160
80 IF I$="H"THEN 120
90 IF I$="T"THEN 140
100 PRINT"NO[1SPC]ES[1SPC]VALIDO
"
110 GOTO 60
120 H=H+1
130 GOTO 60
140 T=T+1
150 GOTO 60
160 PRINT"DE[1SPC]";H+T;"[1SPC]T
IRADAS"
170 PRINT"HAN[1SPC]SALIDO[1SPC]"
;H;"[1SPC]CARAS"
180 PRINT"[9SPC]Y[1SPC]";T;"[1SP
C]CRUCES"
190 END
```

PROGRAMA: REACCION

```
0 REM TIEMPO DE REACCION 64
5 REM COPYRIGHT (C) ANDREW COLIN
 1981
10 POKE53280,0:POKE53281,0:PRINT
"[1CLR][1WHT]"
20 PRINT
30 PRINT"PARA[1SPC]MEDIR[1SPC]SU
[1SPC]"
40 PRINT"TIEMPO[1SPC]DE[1SPC]REA
CCION"
50 PRINT"PULSE[1SPC]UNA[1SPC]TEC
LA"
60 PRINT"Y[1SPC]ESPERE[1SPC]EL[1
SPC]TONO[1SPC]"
70 PRINT"CUANDO[1SPC]LO[1SPC]OIG
A[1SPC]PULSE"
80 PRINT"UNA[1SPC]TECLA[1SPC]TAN
[1SPC]RAPIDO[1SPC]"
90 PRINT"COMO[1SPC]PUEDA"
100 PRINT:PRINT"[7SPC]BUENA[1SPC
]SUERTE"
110 REM ESPERA UNA TECLA
120 GET A$
130 IF A$=""THEN120
140 REM ESPERA UN TIEMPO ALEATOR
IO
143 PRINT
145 PRINT"ESPERE!"
148 PRINT
150 Q=TI+INT(60+301*RND(0))
160 GET A$
170 IFA$<>""THEN 340
180 IF TI<Q THEN 160
190 REM MARCHA DEL TONO
200 POKE 54295,0
202 POKE54296,15
205 POKE 54278,240
207 POKE54277,16
210 POKE 54273,100
212 POKE54276,33
220 X=TI
230 REM ESPERA UNA TECLA
240 GET A$
250 IFA$=""THEN240
260 REM COGER RESULTADO
270 R=TI
280 POKE54276,0
290 POKE54296,0
300 REM MOSTRAR RESULTADO
310 PRINT"SU[1SPC]TIEMPO[1SPC]DE
[1SPC]REACCION[1SPC]ES"
320 PRINT(R-X)/60;"SEGUNDOS
330 STOP
340 PRINT"DEMASIADO[1SPC]PRONTO[
1SPC]"

350 STOP
```

Drean commodore 64 Parte I